1 Automatisierung , Programmierung mit Excel

1.1 Aufbau und Allgemeininformationen

Aufbau und inhaltliche Konventionen des Skriptes

- ✓ Am Anfang jedes Kapitels finden Sie eine kurze Information über die allgemeine Einführung zu dem Programmbeispiel mit wichtigen Funktionen.
- ✓ Die meisten Kapitel enthalten das Programmbeispiel, mit dessen Hilfe praktische Übungen ausgeübt werden können.
- ✓ Das Ziel dieses Skriptes ist einen schnelleren Einstieg in Excel zu ermöglichen.

Typografische Konventionen

Im Text sind drei Schriftarten

- Calibri: für den Schriftsatz des Skriptes
- Adobe Arabic: für alle zugewiesenen Namen wie Abbildungen, Tabellennamen, Formularnamen, Steuerelementen usw.
- `Courier New:` für Programmcodes

angewandt.

Symbole


Besondere Informationen, die Ihnen weiterhelfen können.


Besondere Hinweise oder Tipps


Warnhinweise oder besondere Aufmerksamkeit

Einleitung

Dieses Skript wurde für einen Kurs des Studienganges „Bekleidung- Technik und Management Lehrstuhl Automatische Datenverarbeitung“ erstellt. Deshalb sind die Beispiele darauf zugeschnitten. Die Beispieldateien sowie die in den Übungen verwendeten Übungs- und Ergebnisdateien können Sie im Abschnitt „Aufgaben A1-A5“ vollständig finden.

Außerdem war nicht Vollständigkeit gefragt, sondern es sollte um die Formulare und VBA (*Visual Basic for Application)* der Office-Anwendung Excel 2100 gehen. Grundlegende Kenntnisse wie z.B. Arbeiten mit Arbeitsmappen, Arbeitsblättern, Formeln und Funktionen, sowie Formatieren von Tabellen, Anpassen der Excelkonfigurationen und ähnliches werden hier vorausgesetzt.
In diesem Skript lernen Sie die folgenden Aufgaben zu meistern:

- Hinweise zur Funktionalitäten der Software
- Erstellen und Implementieren von Aufgaben aus den Bereichen Material- und Artikelkalkulation
- Umgang mit den Tabellenblättern
- Umgang mit den Steuerelementen zur Steuerung der Daten

- Umgang mit einer Userform, um die Daten z.B. von einer Stammdatentabelle in eine Ausgabetabelle zu übertragen
- Zugriff auf die Zellen in einer Tabelle
- Umgang mit VBA-Syntax (u.a Variablen und deren Datentypen, Gültigkeitsbereich einer Variable, Fallunterscheidungen, Schleifen, Konstanten, Operatoren, eindimensionalen Arrays, benutzerdefinierten Prozeduren)
- Konzeption und Realisation der Algorithmen zu der Aufgabestellung

Hinweise zu Funktionalitäten

Excel verfügt über einfache Datenbankfunktionen. Es verfügt nicht über die Funktionalitäten des weitverbreiteten Konzepts der relationalen Datenbank. Es kann jedoch über Schnittstellen auf Daten aus Datenbanken zugegriffen werden. Mit Excel 2010 Professional kann wie in einem Datenbanksystem (DBS) ein System zur elektronischen Datenverwaltung implementiert werden. Die wesentliche Aufgabe eines DBS ist es, Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern und benötigte Teilmengen in unterschiedlichen, bedarfsgerechten Darstellungsformen für Benutzer und Anwendungsprogramme bereitzustellen. In diesem Skript behandelter Aufgaben geht es darum, das prinzipielle Konzept des Datenbanksystems zu implementieren. Dabei sind folgende Punkte wichtig:

- Datenbank vorbereiten (Userform mit den notwendigen Steuerelementen zur Verfügung zu stellen, Stammdatentabellen und Ausgabedatentabellen festzulegen).
- Datensätze bearbeiten (Datensatz für den Benutzer zu holen, Datensatz in die Stammdatentabelle zu ersetzen, hinzuzufügen, zu löschen, zu sortieren und zu suchen).
- Datensätze mit den kalkulierten Daten ausgeben (Datenmengen in unterschiedlichen, bedarfsgerechten Darstellungsformen für Benutzer bereitzustellen).
- Softwarekriterien berücksichtigen (Benutzerfreundlichkeit, Korrektheit, Erweiterbarkeit usw.).

1.2 Automatiesierungsmöglichkeiten mit Excel

Welche Arbeiten können automatisiert werden?

Visual Basic for Application, so lautet der ausgeschriebene Name für die Makrosprache der Microsoft Anwendungen. Damit kann man häufig wiederkehrende Befehlsabfolgen automatisieren, Bedingungen einfügen, Schleifen durchlaufen lassen und vieles mehr. Wenn Sie z.B. die Zellen in einer Tabelle, die als Zahl eingestellt sind, als Währung formatiern wollen, können Sie ein Makro einsetzen, um Ihre Arbeit schneller und einfacher zu erledigen.

Makros aufzeichnen

Um ein Makro aufzuzeichnen, brauchen Sie keine VBA-Programmierkenntnisse. Sie benötigen das automatische Aufzeichnen, indem Sie im Register „Ansicht“ in der Gruppe „Makros“ auf den Pfeil des Symbols „MAKROS“ klicken und den Eintrag „MAKRO AUFZCHN.“ in der geöffneten Liste wählen. Nach dem Sie einen Makronamen eingeben haben, wird jeder

Arbeitsschritt, den Sie während der Aufzeichnung des Makros ausführen (z.B. Rahmen und Inhalt der Zellen Formatieren) wird automatisch sogenannter VBA-Code generiert.

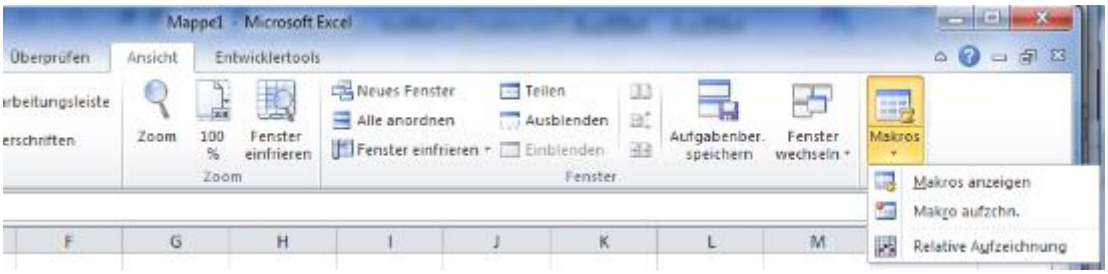


Abb. 1.2.1 Makro aufzchn.

Makronamen festlegen

Ein Makro beinhaltet eine Folge von automatisch abzuarbeitenden Aufgaben, die in Form der VBA-Anweisungen vorkommen. Ein Makro muss einen Namen haben, mit dem die Aufgaben ausgeführt bzw. aufgerufen wird. Wird ein Makro Abb. 1.2.1 Makro aufzchn. Erstellt, erscheint ein Dialogfenster Abb. 1.2.2 Makroeigenschaften festlegen.

- Geben Sie im Textfeld bei *Makroname* einen Namen für Ihr Makro. Dabei sind auf die folgenden Regeln zu beachten:
  - ✓Der Makroname muss mit einem Buchstaben beginnen.
  - ✓Sonderzeichen sind nicht erlaubt.
- Wenn Sie das Makro später über eine Tastenkombination ausführen wollen, drücken Sie eine Taste ins Textfeld bei *Tastenkombination*.
- Achten Sie darauf, wo Ihr Makro gespeichert wird. Meistens will man das Makro in derselben Arbeitsmappe speichern. Wählen Sie Ihre Auswahl im Kombinationsfeld bei *Makro speichern in*:

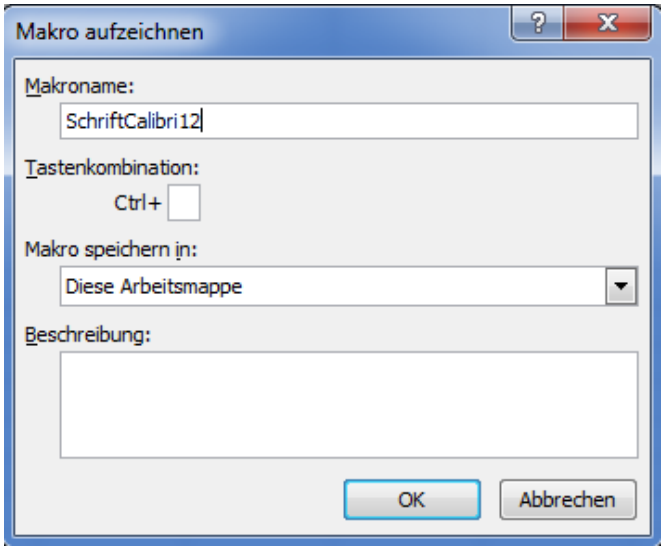


Abb. 1.2.2 Makroeigenschaften festlegen

- Zweck des Makros wird in das Textfeld bei *Beschreibung* eingetragen.
- Anschließend bestätigen Sie die Schaltfläche OK, um Ihre Makroaufzeichnung zu starten.

Arbeitsschritte aufzeichnen

Nachdem Sie in Abb. 1.2.2 Makroeigenschaften festlegen auf OK geklickt haben, wird Excel im Aufzeichnungsmodus befindet und erscheint das Symbol ■ in der Statusleiste.

Wenn Sie ein Makro aufzeichen und beibehalten wollen, müssen Sie die entsprechende Arbeitsmappe speichern. Während der Aufzeichnung dürfen Sie den Programmbereich nicht verlassen. Dabei wird jeder Arbeitsschritt aufgenommen.

Makros müssen in Excel 2010 im Dateiformat EXCEL-ARBEITSMAPPE MIT MAKROS gespeichert werden. Als Dateinamenerweiterung wird .xlsm verwendet.

Wenn Sie die Arbeitsschritte nicht genau kennen, testen Sie vorher ohne Aufzeichnung.

Makroaufzeichnung beenden

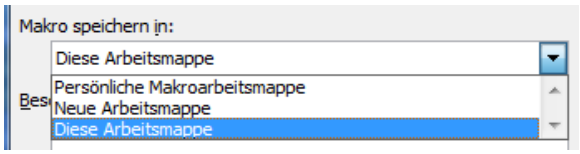
Zum Beenden Ihres Makros können Sie entweder

- Auf das Symbol klicken oder
- Im Register ANSICHT in der Gruppe MAKROS in der Liste auf den Eintrag AUFZEICHNUNG BEENDEN klicken.

Makro speichern

Wenn Sie ein Makro aufzeichen wollen, achten Sie darauf, wo das Makro gespeichern wird.

Makros werden standardmäßig innerhalb einer Arbeitsmappe gespeichert. Wie es im Abschnitt „Makronamen festlegen“ beschrieben wurde.



Sollte der Makroname in der Arbeitsmappe existieren, meldet Excel ein Fenster (Abb. 1.2.3), in dem Sie aufgefordert werden, zu ersetzen.

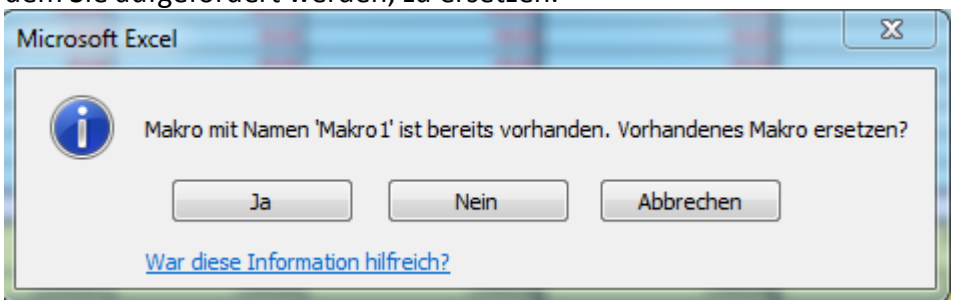


Abb. 1.2.3 Rückfrage zum Speichern von Makros in der persönlichen Makroarbeitsmappe

Damit Ihr Makros entgültig in der Arbeitsmappe gespeichert wird, gehen Sie wie folgt vor:

- wählen Sie in Menü DATEI den Menüpunkt SPEICHERN UNTEN. Darauf erscheint das Fenster (Abb. 1.2.4).
- Wählen Sie für den Dateityp „Excel-Arbeitsmappe mit Makros.
- Geben Sie im Listenfeld DATEINAME der gewünschten Dateinamen ein.
- Bestätigen Sie mit der Schaltfläche SPEICHERN.

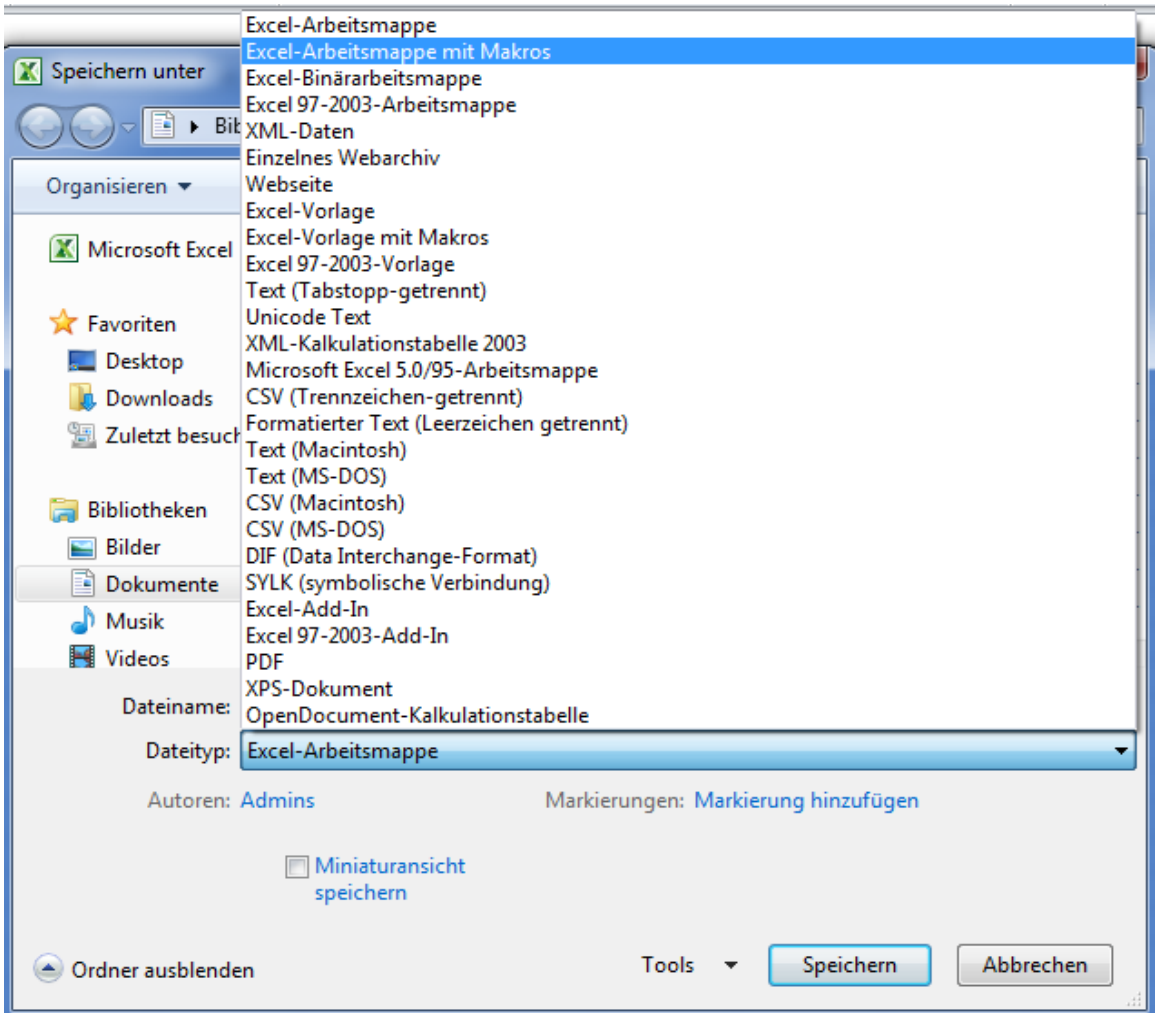


Abb. 1.2.4 Dialogfenster zum Datei speichern

Makros ausführen, löschen und anzeigen

Beim Ausführen von Makros müssen Sie darauf achten, dass Makros aus vertrauenswürdigen Quellen entstanden wurde. Denn Makros können Viren enthalten. Standardmäßig werden Makros in Excel unterdrückt, wenn Sie eine Arbeitsmappe mit gespeicherten Makros öffnen. In diesem Fall erhalten Sie eine Sicherheitswarnung (Abb. 1.2.5), wenn Sie noch nicht die Einstellungen für Makros auf ALE MAKROS AKTIVIEREN (Abb. 1.2.6) eingestellt haben.

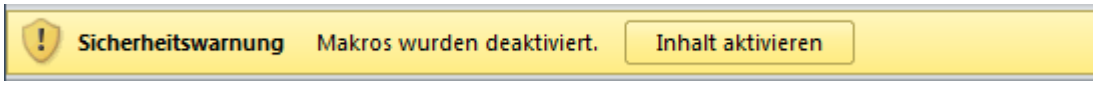


Abb. 1.2.5 Sicherheitswarnung für die Makroaktivierung

Bestätigen Sie die Schaltfläche INHALT AKTIVIEREN nur, wenn Sie der Herkunft der Makros vertrauen. Nach Bestätigung wird die Datei dauerhaft zu einem vertrauenswürdigen Dokument erklärt und die nachfolgend beschriebenen Wahlmöglichkeiten stehen Ihnen dann nicht mehr zur Verfügung.

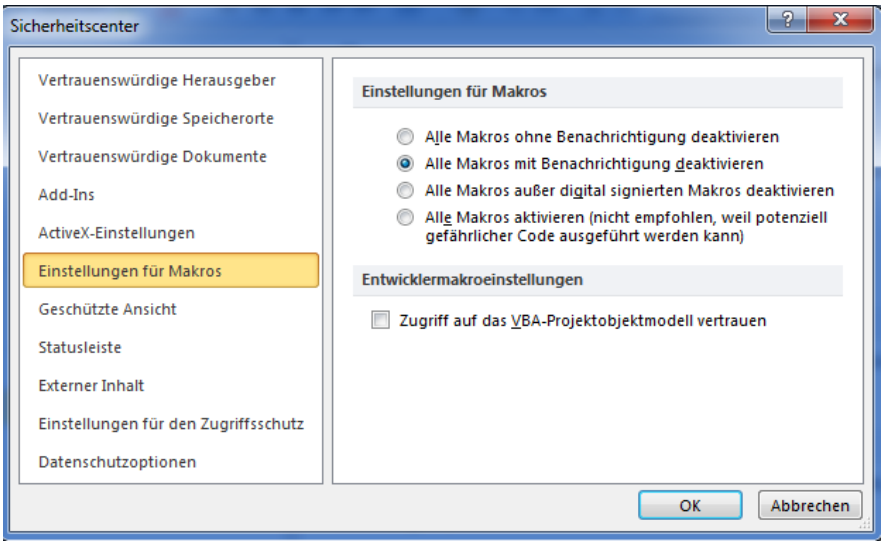


Abb. 1.2.6 Sicherheitswarnung für die Makroaktivierung

Wenn Sie der Herkunkt der Makros vertrauen, können Sie auf das Optionsfeld ALLE MAKROS AKTIVIEREN (Abb. 1.2.6) klicken. Um zu diesem Optionsfeld gelingen, gehen Sie wie folgt vor:

- Wählen Sie WEITERE BEFEHLE in SYMBOLLEISTE FÜR DEN SCHNELLZUGRIFF ANPASSEN.
- Wählen Sie dann im Menüpunkt SICHERHEITSCENTER die Schaltfläche EINSTELLUNGEN FÜR DAS SICHERHEITSCENTER.
- Klicken Sie anschließend auf den Menüpunkt EINSTELLUNGEN FÜR MAKROS und wählen Sie das Optionsfeld ALLE MAKROS AKTIVIEREN.

Nach dem Sie Makroseinstellungen vorgenommen haben, können Sie im Register ANSICHT in der Gruppe MAKROS auf den Pfeil des Symbols MAKROS klicken und den Eintrag MAKRO ANZEIGEN in der geöffneten Liste wählen. Dann erscheint das Dialogfenster (Abb. 1.2.7).

In diesem Dialogfenster haben Sie die Möglichkeiten, ein Makro ausführen, löschen oder auch bearbeiten bzw. anzeigen.

Wenn Sie ein Makro auswählen und auf die Schaltfläche AUSFÜHREN klicken, wird das Makros ausgeführt. Sollte ein Fehler in Makro vorkommen, wird der Fehler im Editor (Siehe im Abschnitt „der Visual Basic Editor“) angezeigt.

Wenn Sie ein Makro löschen wollen, klicken Sie auf die Schaltfläche LÖSCHEN. Das Makro wird aus Ihre Arbeitsmappe für immer gelöscht, wenn Sie Ihre Arbeitsmappe dann speichern.

Abb. 1.2.7 Dialogfenster MAKRO ANZEIGEN

Wenn Sie die Codes eines Makros anzeigen und bearbeiten wollen, klicken Sie auf die Schaltfläche BEARBEITEN. In diesem Fall wird der Visual Basis Editor angezeigt.

2 Grundlagen zum Visual Basic-Editor

2.1 Aufrufen und Einrichten

In allen Microsoft Anwendungen, also auch in Excel, kann man mit der Tastenkombination ALT + F11 den Editor aufrufen. Alternativ geht das auch mit der Befehlsfolge **Extras *Makro*** *Visual Basic Editor*. Es öffnet sich ein *eigenes Fenster*, das nicht Bestandteil des Excel-Bildschirms ist. Das Bild, das sich dann bietet, ist noch nicht besonders arbeitsfreundlich, deshalb sollte man etwas über die drei wichtigsten Teile des Fensters wissen.

Der ***Projekt-Explorer*** sitzt meistens oben links in der Ecke und zeigt wie in einem Verzeichnisbaum die geöffneten Dateien, wobei jede für ein Projekt steht. Unterhalb dieser Ebene sieht man die jeweils zugehörigen Objekte. Das können Module sein, Formulare, Klassenmodule oder auch Tabellenblätter. Jedes Objekt aus Tabellen und Formularen kann aus den Objektteil und Programmcode bestehen. In diesem Skript werden hauptsächlich Programmcodes in Formularen behandelt.

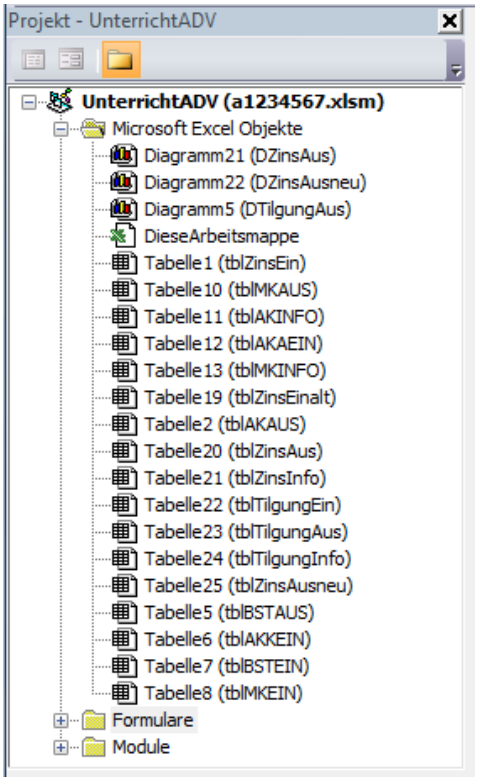


Abb. 2.1.1Projekt-Explorer

Das ***Eigenschaftsfenster*** sitzt direkt darunter und zeigt die verschiedenen einzustellenden Eigenschaften der jeweiligen Objekte an. Die Graphik zeigt die Eigenschaften eines Formulars, wobei man hier mal das erste Fremdwort lernen sollte, nämlich USERFORM. So heißen diese Objekte in VBA. Falls eines dieser beiden Fenster nicht zu sehen ist, kann man sie über **Ansicht *Projekt-Explorer*** bzw. ***Eigenschaftsfenster*** anschalten

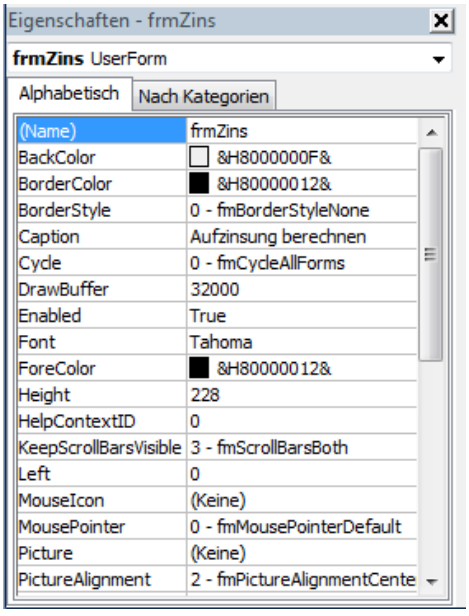


Abb. 2.1.2 Eigenschaftsfenster

Rechts davon, den größten Teil des Bildschirms (Abb. 2.1.3) einnehmend, findet das sogenannte ***Modul-fenster*** seinen Platz. Je nach dem, was man als Objekt im Projekt-Explorer markiert hat, sieht man dort Programmcode oder ein User form (Objektteil).

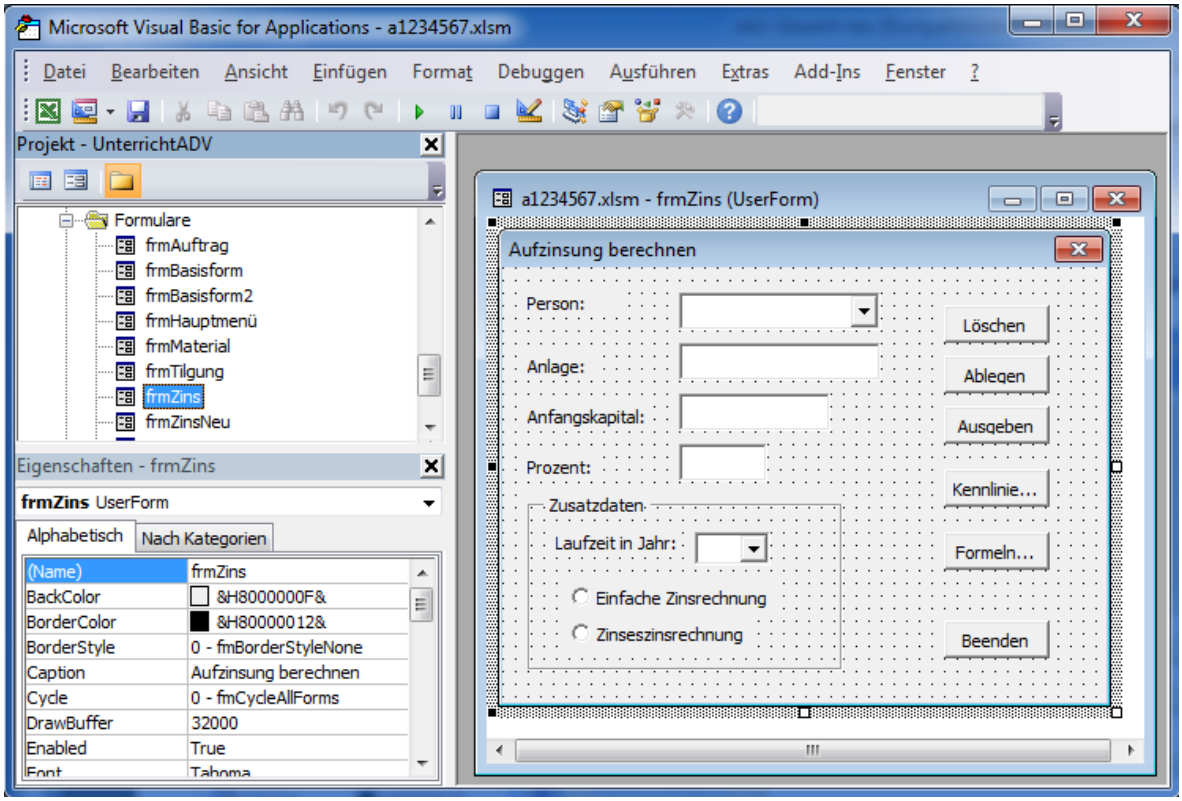


Abb. 2.1.3 Objektansicht von der Userform z.B. „frmZins“

Bei diesem Beispiel ist links die UserForm mit dem Namen „frmZins“ markiert und rechts ist es als Graphik zu sehen.

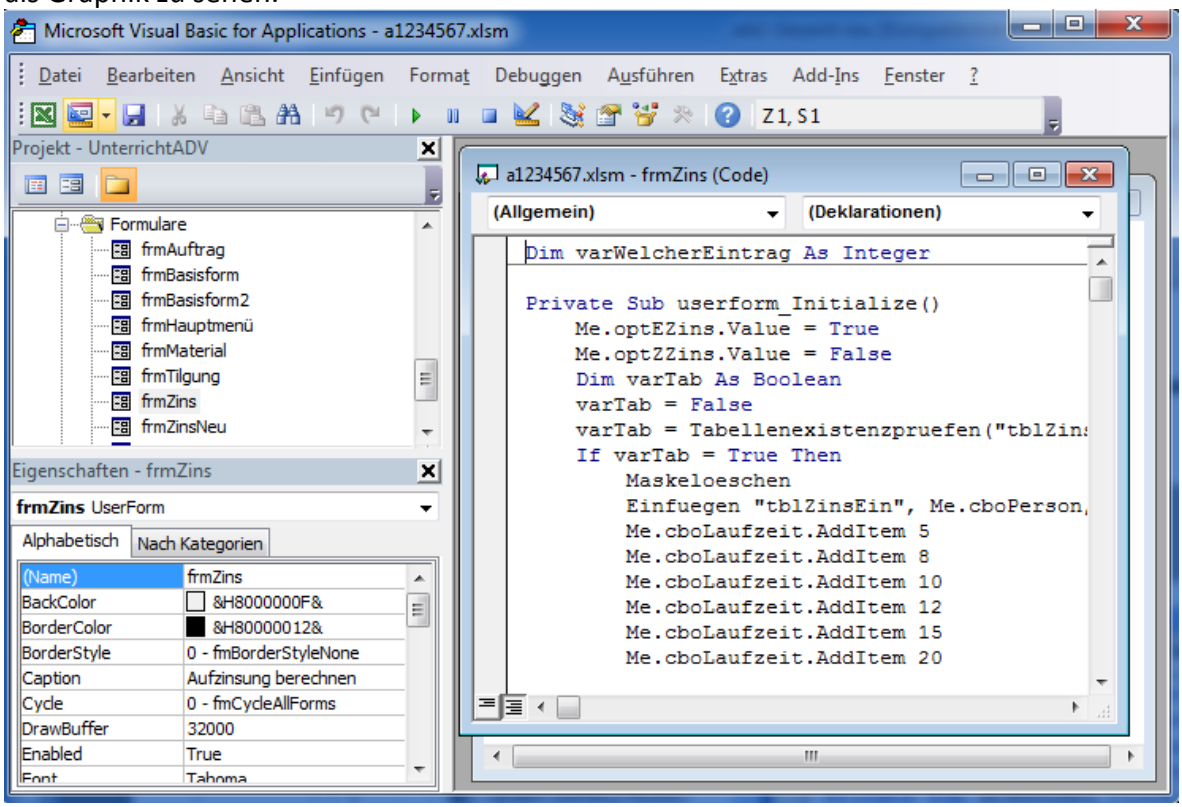
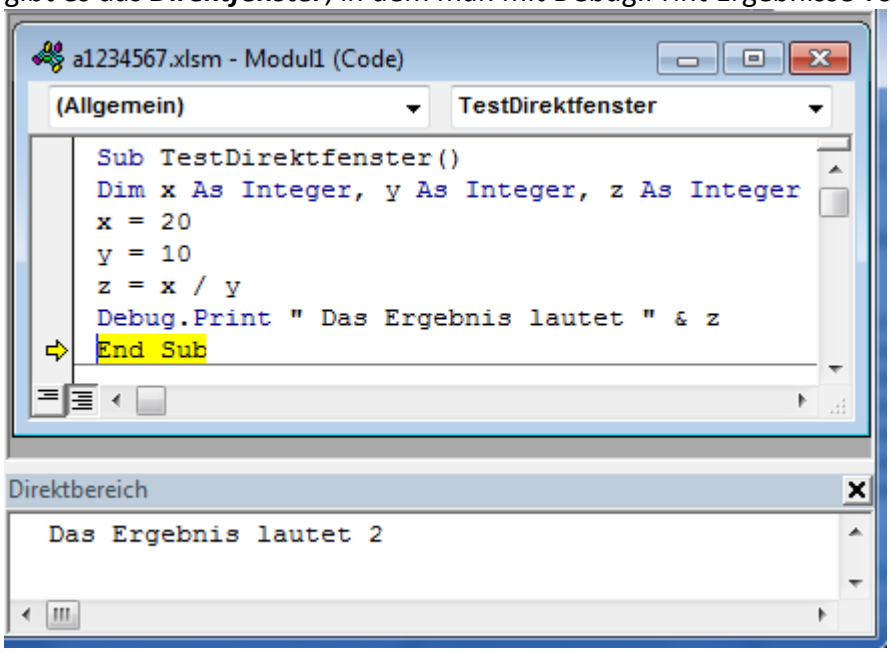


Abb. 2.1.4 Codeansicht von der Userform z.B. „frmZins“

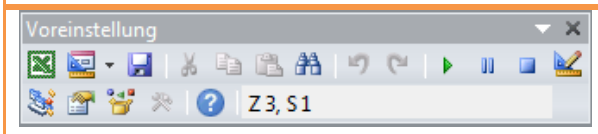
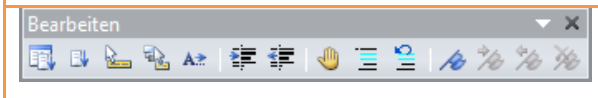
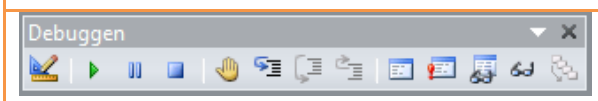
In dieser Ansicht ist links ein Modul markiert und rechts wird der dort abgelegte Code angezeigt und später dann natürlich auch eingegeben. Zwei weitere Fenster sind noch interessant, allerdings hat man sie in den wenigsten Fällen immer eingeschaltet. Zum einen gibt es das ***Direktfenster***, in dem man mit Debug.Print Ergebnisse von Variablen ausdrucken...



Das ***Lokalfenster*** dient zur direkten Überprüfung von Variablen. Es liefert zu allen globalen und lokalen Variablen des aktuellen Moduls den Wert und den Datentyp, wenn Sie mit ***F8*** durch den Code hoppeln.

 Beim halten der direkten Überprüfung klicken Sie auf das Symbol ZURÜCKSETZEN  in der Symbolleiste VOREINSTELLUNG.

Symbolleisten des Visual Basic-Editors

Symbolleiste	Beschreibung
	Die häufigsten verwendeten Befehle befinden sich in dieser Symbolleiste. Die Befehle wie Makros starten, beenden, Zurücksetzen, Einblenden des Objektkatalogs die Rückgängigfunktionen, Einfügen von Objektelementen und die Suchfunktion.
	Enthält häufig verwendete Befehle zum Berarbeiten von Quellcode
	Enthält Befehle für das Testen von Visual Basic-Programmcode, z.B. Makro Starten, unterbrechen und beenden, Einzelschrittmodus, Anzeige von Lokal- und Direktfenster.
	Enthält Befehle, mit den Sie die Steuerelemente in den selbst erstellten Dialogfenstern positionieren können.

2.2 Makros bearbeiten und verwalten

Aufbau des Makro-Codes

Der Makro-Code besteht aus Visual Basic-Anweisungen, die beim Ausführen des Makros nacheinander abgearbeitet werden. Ein aufgezeichnetes Makros hat den folgenden Aufbau:

```
Sub TextfarbeRot()
'
' TextfarbeRot Makro
'
' Tastenkombination: Strg+Umschalt+S
'
    With Selection.Font
        .Color = -16776961
        .TintAndShade = 0
    End With
End Sub
```

1. Jedes Makro wird mit dem Schlüsselwort `Sub` (für engl. Subroutine = Unterprogramm) eingeleitet. Anschließend folgt der Name des Makros und ein Rundenklammernpaar. Man nennt hier die Deklaration eines Makros.
2. Hochkomma-Zeichen bedeutet für Visual Basic for Application Kommentare. Kommentare haben keinen Einfluss auf die Ausführung des Makrocodes und dienen nur der Beschreibung des Programmcodes.
3. Mit der `With … End With` – Anweisung wird eine vereinfachte Schreibweise ermöglicht. Hier beziehen sich die in der `With`-Anweisung eingeschlossenen Anweisungen alle auf das Objekt `Selection.Font`. In der `With`-Anweisung kann deshalb z.B. `.Color` anstelle von `Selection.Font.Color` verwendet werden.
4. Das Ende des Makros ist durch die Schlüsselwörter `End Sub` gekennzeichnet.

Module bzw. Makros kopieren

Makros werden in Module gespeichert. Sie können nur Module aus Arbeitsmappen kopieren, die geöffnet sind. Um einmal erstellte Makros auch in anderen Arbeitsmappen benutzen zu können, wird das entsprechende Modul, welches die Makros enthält dorthin kopiert. Mit dem Modul kopieren Sie alle darin enthaltenen Makros bzw. VBA-Programmmodule.

Module exportieren

Sie haben die Möglichkeit, ein Modul als selbstständige Datei zu speichern. Diese Datei lässt sich später in andere Arbeitsmappen importieren. Abhängig von der Art des Moduls werden verschiedene Dateinamenserweiterungen verwendet: Standardmodule erhalten die Endung **.bas*, Formulamodule **.frm* und Klassenmodule **.cls*.

Module importieren

Ein exportiertes Modul in Form einer Datei können Sie in die im Projekt-Explorer ausgewählte Arbeitsmappe einfügen bzw. importieren.

2.3 Aufruf eines Makros über Symbolleisten

Für den Aufruf eines Makros über eine Symbolleiste ist eine Schaltfläche in einer der vorhandenen oder einer neuen Symbolleiste nötig. Die Erstellung und Zuordnung ist in folgenden Schritten möglich:

- Das Dialogfenster (Abb. 2.3.1) DATEI/OPTIONEN wählen.
- Im Dialogfenster EXCEL-OPTIONEN MENUBAND ANPASSEN wählen.
- Beim BEFEHLE WÄHLEN MAKROS auswählen.
- AUF die Schaltfläche NEUE REGISTERKARTE klicken.
- Registerkarte nach einem gewünschten Namen z.B. ADV-PROGRAMME umbenennen.
- Neue Gruppe in dieser Registerkarte nach einem gewünschten Namen z.B. HAUPTMENÜ umbenennen.
- Ihren Makronamen z.B. MCBANDEGANI auswählen und auf die Schaltfläche HINZUFÜGEN>> klicken.
- Mit der rechten Maus auf das Symbol in der Registerkarte MCBANDEGANI klicken, und Menüpunkt UMBENNEN auswählen. Dann erscheint ein Dialogfenster (Abb. 2.3.2).
- Ein Symbol nach Ihrer Wahl dem Makro zuordenen.

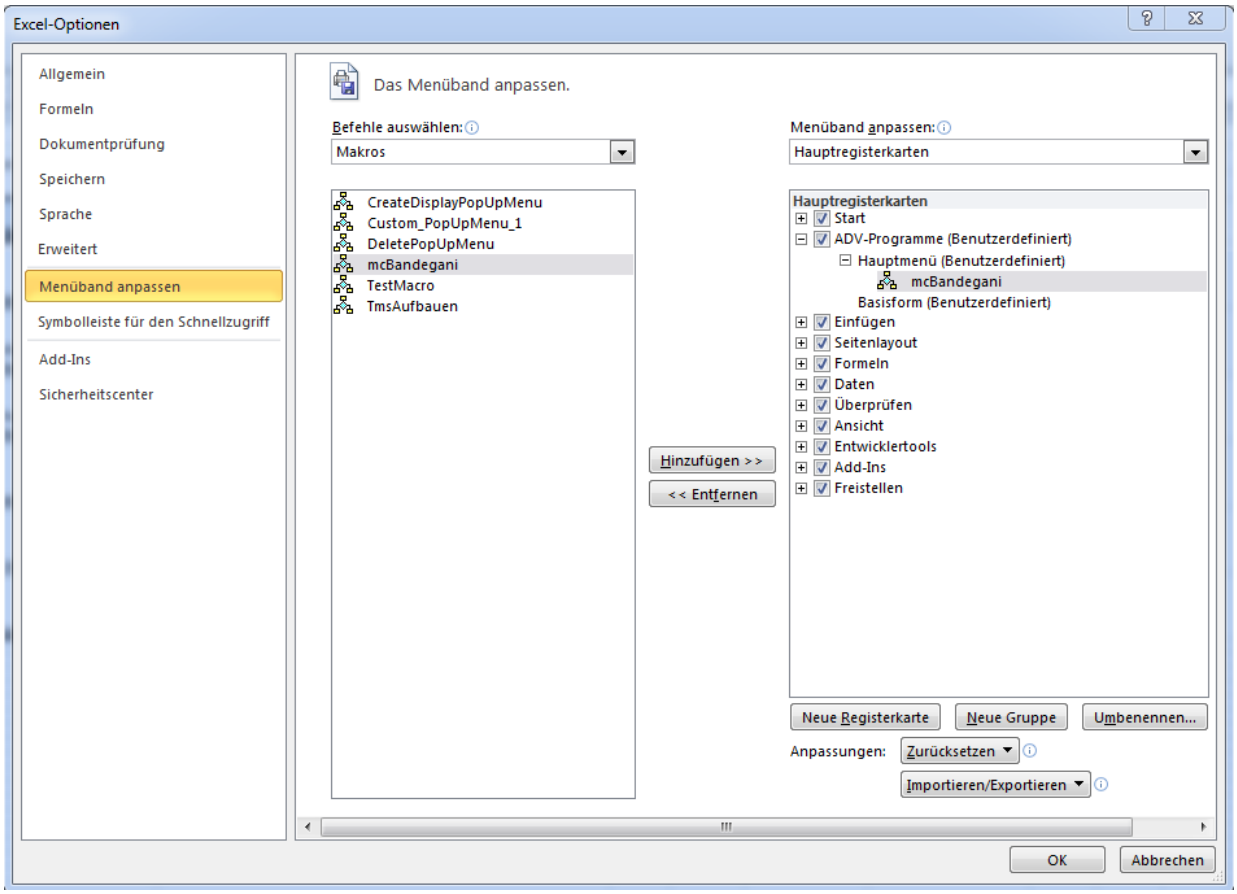


Abb. 2.3.1 Excel-Optionen zur Einrichtung der Registerkarte

Wenn Sie mit allen Schritten fertig sind, bestätigen Sie Ihre Einstellung mit der Schaltfläche OK. Dann generiert Excel eine Menüleiste mit den Gruppennamen und darunter Symbol bzw Symbole für Ihre Makros (Abb. 2.3.3).

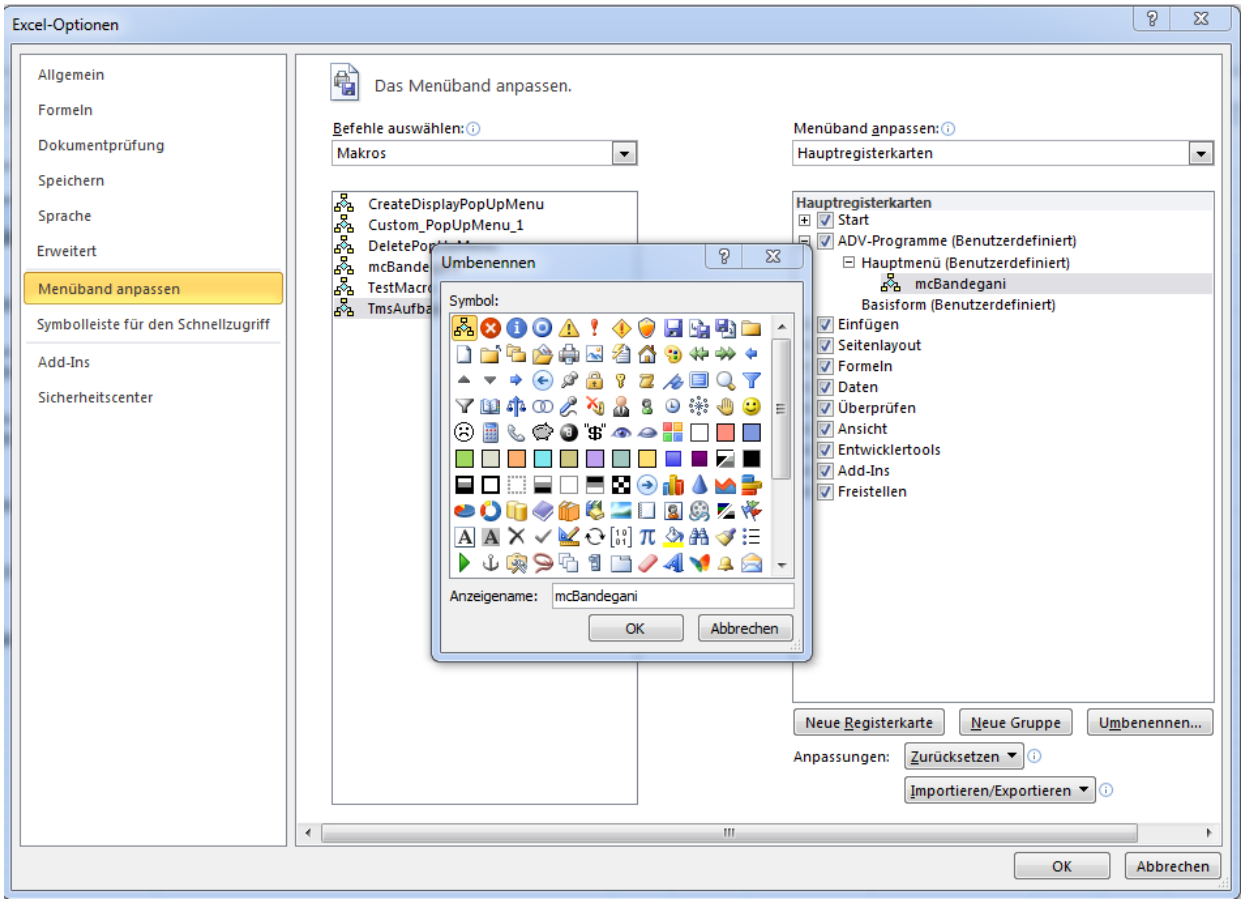


Abb. 2.3.2 Zuordunung eines Symbols für ein Makro in der Registerkarte

Abb. 2.3.3 Generierte Register, Gruppe und Makros

Registerkarte gehört zu der Microsoft Office Benutzeroberfläche. Daher wird diese nur auf einem Rechner sichbar, auf dem Sie eingerichtet haben.

3 VBA Grundlagen

VBA – *Visual Basic for Applications* ist eine objektorientierte Makroprogrammiersprache mit einer sehr umfangreichen Auswahl von Funktionen und Anweisungen zur Erstellung eigenständiger Programme, die MS Office – Anwendungen automatisieren und oft aus der Sicht des Anwenders komplexe Abläufe in Ihrer Bedienung vereinfachen.

VBA – Module unterscheiden sich je nach Office – Produkt in Ihrer Struktur, hier sollen die Spezifika des Excel – VBA in ihren Grundrissen vorgestellt werden. VBA – Module sind vom Sprachumfang her in zwei inhaltliche Bereiche teilbar:

- Den die Programmstruktur definierenden *Sprachkern*, der mit dem von VB (Visual Basic) gleichzusetzen ist, mit Schlüsselwörtern, Befehlen, Funktionen und Kontrollstrukturen, die für alle Office – Anwendungen gelten.
- Die anwendungsspezifischen *Objektmodelle*, mit eigenen, den Anwendungen angepaßten OBJEKTEN, EIGENSCHAFTEN, EREIGNISSEN und METHODEN.

Da insbesondere diese vier Begriffe immer wieder als Grundbegriffe der objektorientierten Programmierung im Zusammenhang mit VBA auftreten und somit auch im Inhalt dieser Unterlage öfter vorkommen, sollen sie wie folgt vorgestellt werden.
OBJEKTE stellen einen zentralen Bestandteil fast aller VBA – Anwendungen dar. Sie sind bestimmte Teile einer Anwendung – Excel selbst, die Arbeitsmappe, das Tabellenblatt, Zellen (-bereiche). Objekte besitzen EIGENSCHAFTEN (benannte Attribute), beispielsweise Größe, Farbe, Position, Name usw.
Das Verhalten eines Objekts wird über die METHODEN (= Prozeduren), die auf ein Objekt angewandt werden, bestimmt.

3.1 Objekte und ihre Hierarchie

Excel besteht aus mehr als 200 Objekten, deren Eigenschaften und Methoden frei zugänglich sind. Zu den wichtigsten Objekten zählen:

APPLICATION	Excel selbst / Excel – Fenster
WORKBOOK	Die Excel – Arbeitsmappe
WORKSHEET	Ein Tabellenblatt
RANGE	Zellenbereich , bestehend aus einer oder mehrerer Zellen

Die Objekte stehen in hierarchischer Abhängigkeit zueinander. Objekte höherer Stufe beinhalten Objekte untergeordnet Stufe(n). Für o.g. Objekte gilt die Hierarchieordnung.
Die gesamte Objekthierarchie (Abb. 3.1) ist natürlich wesentlich komplizierter und umfangreicher.

Abb. 3.1 Objekthierarchie von Excel (Ausschnitt)

Für die Programmierung ist es wichtig zu wissen, daß die Objekthierarchie nicht nur ein willkürliches Ordnungsschema ist. Die Kenntnis der festgelegten Position eines Objekts in der Hierarchie ist nötig, um das Objekt zu referenzieren – darauf zuzugreifen.
Beispiel:

```
Worksheets("Tabelle1").Range("A7")
```

Eine besondere Form von Objekten bilden die AUFLISTUNGEN (Collections). Es sind Gruppen gleichartiger Objekte, zusammengefaßt in einem Container – der *Auflistung*, z.B. die Auflistung WORKSHEETS, die alle Tabellenblätter einer Arbeitsmappe enthält.

3.2 Eigenschaften

Während ein Objekt weitgehend abgeschlossen ist, d.h. sein innerer Aufbau lässt sich i.d.R. nicht beeinflussen, lassen sich seine EIGENSCHAFTEN per Programmanweisung in vielen Fällen verändern. Die Veränderung der Eigenschaften nimmt oft einen wesentlichen Teil eines Programms ein.

Eigenschaften sind benannte Attribute eines Objekts. Sie bestimmen seine Charakteristika wie Größe, Farbe oder Bildschirmposition, aber auch den Zustand, wie beispielsweise *aktiviert* oder *deaktiviert*.
Es gibt Eigenschaften, die lesbar und veränderbar sind, z.B. *Value* (Wert) oder *Name* (Name), andere lassen sich nur abfragen, aber nicht verändern – es sind sog. *Nur – Lese Eigenschaften*.
Zu den Eigenschaften, die besonders oft verändert werden, gehören:

CAPTION	Beschriftungen von Objekten
NAME	Die Bezeichnung eines Objekts, unter der es angesprochen (referenziert) wird.
SELECTION	Bezeichnet das markierte Application – Objekt.
VALUE	Wert / Inhalt eines Objekts (Zellinhalt, Textfeld – Eintrag)

Alle Eigenschaften haben immer einen aktuellen Wert. Die Anzahl möglicher Werte ist unterschiedlich groß. So besitzen beispielsweise die Eigenschaften *Color* oder *Value* sehr viele Werte, während andere, beispielsweise *Selected*, nur die Werte *False* oder *True* annehmen.
Beispiel:

```
Worksheets("Tabelle1").Range("A3:D7").Value = 55
```

Die Anweisung weist allen Zellen des Zellbereichs A3:D7 des Tabellenblatts *Tabelle1* den Wert *55* zu.

3.3 Methoden

Zu den Objekten gehören neben Eigenschaften auch METHODEN. Über Methoden läßt sich das Verhalten von Objekten steuern / verändern. Eine Methode ist eine Aktion, die eine Operation auf einem Objekt ausführen kann. Zu den am häufigsten benutzten Methoden gehören:

OPEN	öffnet eine Arbeitsmappe
CLOSE	schließt eine Arbeitsmappe (ein Workbook-Objekt) oder Excel (ein Application-Objekt).
CLEAR	löscht einen Zellbereich oder ein Diagramm.
AKTIVATE	aktiviert ein Objekt
SELECT	wählt ein Objekt aus

Für den Einsteiger ist es oft problematisch zwischen Eigenschaften und Methoden zu unterscheiden (insbesondere wenn Methoden die gleichen Namen tragen wie beispielsweise

Auflistungen). So sind beispielsweise die beiden Anweisungen:

```
Assistant.Visible = true
Assistant.Move 250,275
```

zumindest optisch sehr ähnlich. Es stellt sich die Frage - sind `VISIBLE` und `MOVE` Eigenschaften oder Methoden und wenn nicht welches Element von beiden ist eine Methode und welches ein Eigenschaft.
Die Antwort ist hier relativ einfach: Bei Zuweisungen von Eigenschaften wird das Gleichheitszeichen benutzt, bei Methoden benutzt man (optionale) Parameter ohne Gleichheitszeichen.
Bei der Programmierung in der VBA – Umgebung wird die Entscheidung Eigenschaft /Methode zusätzlich durch spezifische Symbole in entsprechenden Auswahlfenstern erleichtert (s. weiter im Text).
Übrigens: Beide Anweisungen beziehen sich auf den Assistenten „Karlchen Klammer“ – die erste mach ihn sichtbar, die zweite verschiebt Ihn entsprechend den angegebenen Parametern.

3.4 Ereignisse

Für die meisten Objekte sind explizite Ereignisse definiert, denen fest zugeordnete *Ereignisprozeduren* zugehören. Ereignisse können Aufrufe von Menüfunktionen, Anklicken von Schaltern, Symbolen und Tasten aber auch Öffnen von Dokumenten, Berechnungen, Veränderungen von Inhalten usw. Allgemein gesehen - Mausklicks, Tastatureingaben und systeminterne Aktionen lösen Ereignisse aus, auf die über entsprechende Prozeduren reagiert werden muß.

3.5 Die VisualBasic-Syntax

In diesem Teil wollen wir uns schrittweise und systematisch mit der Syntax von VisualBasic vertraut machen. Die Syntax informiert Sie, welche Sprachelemente verfügbar sind und wie sie syntaktisch richtig eingesetzt werden. Wir beginnen mit den lexikalischen Elementen. VisualBasic besitzt Grammatik- und Zeichensetzungsregeln, die festlegen, welche Zeichen und Wörter einen Sinn geben und in welcher Reihenfolge sie geschrieben werden können.

Beispiel 1)

```
varEndwert = Worksheets("tblZinsEIN").Cells(1, 1).CurrentRegion.Rows.Count
```

Die Anzahl der Zeilen in dem aktuellenbereich angefangen von Zeile 1 und Spalte 1 in der Tabelle *tblZinsEIN* wird ermittelt und der Variable *varEndwert* zugewiesen.

Punkt-Syntax

In VisualBasic verweist ein Punkt (.) auf die zu einem Objekt gehörigen Eigenschaften und Methoden. Ein Punkt-Syntax-Ausdruck beginnt mit dem Namen des Objektes, gefolgt von einem Punkt, und endet mit der Eigenschaft, Methode oder Variablen, die Sie angeben möchten.

Beispiel 2) `Count` ist eine Eigenschaft. Diese Eigenschaft zählt automatisch die Anzahl von dem gesamten Bereich der ersten Spalte in der Tabelle tblZinsEin.

```
Dim varEndwert As Integer
```

```
varEndwert = Worksheets("tblZinsEIN").Cells(1, 1).CurrentRegion.Rows.Count
```

Beispiel 3) In diesem Beispiel wird das Kombinationsfeld cboPerson, das sich im Dialogblatt (Userformblatt) frmZins befindet, mit den Inhalten von der ersten Spalte in der Tabelle tblZinsEin ausgefüllt. Der Index i zählt dabei die Anzahl der Eintragungen. `AddItem` ist eine Methode auf das Objekt DropDowns mit dem Namen cboPerson.

```
Dim varEndwert As Integer
Dim i As Integer
varEndwert = Worksheets("tblZinsEin").Cells(1, 1).CurrentRegion.Rows.Count
i = 0
For i = 1 To varEndwert
   frmZins.cboPerson.AddItem Worksheets("tblZinsEin")Cells(i, 1).Value
Next i
```

Runde Klammern

1. Bei der Definition einer Prozedur (`sub`, `Funtion` und `Property`) werden die Argumente in runde Klammern eingeschlossen

Beispiel 4) Definition von Sub-Prozedur Einfuegen

```
Sub Einfuegen()
  Dim varEndwert As Integer
  Dim i As Integer
  varEndwert = Worksheets("tblZinsEin").Cells(1, 1).CurrentRegion.Rows.Count
  i = 0
  For i = 1 To varEndwert
     frmZins.cboPerson.AddItem Worksheets("tblZinsEin")Cells(i, 1).Value
  Next i
End Sub
```

2. Beim Aufruf einer Funktion werden die Argumente der Funktion in runden Klammern übergeben, beispielsweise:

Beispiel 5) In dem folgenden Beispiel wird die `MsgBox`-Funktion mittels benannter Argumente aufgerufen und der Rückgabewert der Variablen Ergebnis zugewiesen:

```
Mldg = "Wollen Sie die alte Daten Ersetzen?"
Stil = vbYesNo + vbCritical + vbDefaultButton2
Title = "Datensatz Ersetzen"
Ergebnis = MsgBox(Mldg, Stil, Title)
```

3. Sie können mit runden Klammern auch die Reihenfolge von Operationen in VisualBasic beeinflussen oder VisualBasic-Anweisungen leichter lesbar machen.

Beispiel 6)

```
Rem Das Ergebnis ist 14 ohne runde Klammer.
Sum = 2 + 4 * 3
```

Wenn die Addition jedoch geklammert wird, führt VisualBasic zuerst die Addition aus:

Beispiel 7)

```
Rem Das Ergebnis ist  hier  18.
Sum = (2 + 4) * 3
```

Nähere Informationen finden Sie im Abschnitt „Vorrang des Operators“.

4. Runde Klammern dienen auch zum Auswerten eines Ausdrucks. In der folgenden Anweisung bewirken die runden Klammern beispielsweise, dass zwei Parameter an der

Funktion übergeben werden. Die Funktion addiert die beiden Parameter und gibt das Ergebnis an der Funktion weiter:

Beispiel 8)

```
Rem  Funktionsdeklaration
Function zweizahlAddieren(Dim var1 As Integer, Dim var2 As Integer)
    zweizahlAddieren = var1 + var2
End Function

Rem Aufrufen von zweizahlAddieren.
huf=97, mwzhu=2
Print zweizahlAddieren(huf, mwzhu)
Rem Das Ergebnis ist 99 und wird angezeigt.
```

Groß- und Kleinbuchstaben

VisualBasic unterscheidet zwischen Groß- und Kleinschreibung. Die folgenden Anweisungen sind nicht gleichwertig:

Beispiel 9)

```
Dim meineVar As String
meineVar = "Christine"            'richtig
meinevar = "Christine"            'falsch
```

Sie sollten sich konsistente Großschreibungskonventionen angewöhnen, beispielsweise die in diesem Manuskript verwendeten, um Funktions- und Variablennamen beim Lesen von VisualBasic-Code leichter zu erkennen. Wenn Sie bei Schlüsselwörtern nicht korrekt zwischen Groß- und Kleinschreibung unterscheiden, ist Ihr Skript fehlerhaft, das wird dann rot angezeigt. Die richtig geschriebenen Schlüsselwörter werden blau angezeigt. Weitere Informationen erhalten Sie unter Schlüsselwörter und Hervorheben und Überprüfen der Syntax.

Kommentare

Im VisualBasic Modus können Sie den Codes mit Hilfe einer Kommentar-Anweisung Anmerkungen hinzufügen, um die gewünschten Ergebnisse der Aktion festzuhalten. Kommentare erleichtern auch die Weitergabe von Informationen, wenn in einem Entwicklerteam gearbeitet oder Beispiele weitergegeben werden.

Wenn Sie die Kommentare eingeben, müssen Sie davor das Zeichen ' einfügen. Selbst ein einfaches Skript ist leichter verständlich, wenn Sie bei der Erstellung Anmerkungen einfügen:

Beispiel 10)

```
' Ereignis-Click auf die Schaltfläche Datensatz löschen
Private Sub cmdloeschen_Click()
 ' Ist Inhalt vom Kombinationsfeld „cboPerson“ Leer?
 If Me.cboPerson.Text = "" Then
     MsgBox "Sie haben keinen Eintrag gewählt!"
     Beep
     Exit Sub
 Else
     Meldung = "Wollen Sie wirklich den Datensatz löschen?"
     Titel = "Datensatz löschen"
```

```
      Stil = vbYesNo + vbCritical + vbDefaultButton2
      Ergebnis = MsgBox(Meldung, Stil, Titel)
      ' hat der Benutzer auf die Schaltfläche JA geklickt?
      If Ergebnis = vbYes Then
          Worksheets("tblZinsEin").Rows(varWelcherEintrag).Delete Shift:=x1up
          ' die Makros Maskeloeschen und Einfuegen aufrufen.
          Maskeloeschen
          Einfuegen
      Else
          Exit Sub
      End If
  End If
End Sub
```

Kommentare werden im Codefenster in der Farbe Grün angezeigt. Sie können beliebig lang sein, ohne dass die Größe der exportierten Datei dadurch beeinflusst wird, sie unterliegen nicht den Regeln bezüglich VisualBasic-Syntax oder Schlüsselwörtern, und sie können neu in jeder Zeile eingefügt werden.

Schlüsselwörter

In VisualBasic sind einige Wörter für eine bestimmte Verwendung reserviert. Diese können daher nicht als Namen für Variablen, Funktionen oder Labels verwendet werden. In der folgenden Tabelle sind einige VisualBasic-Schlüsselwörter aufgeführt:

• Ablaufsteuerung-Schlüsselwörter (Zusammenfassung)	
Aktion	**Schlüsselwörter**
Ausführen einer Schleife	`Do...Loop, For...Next, For Each...Next, While...Wend, With`
Beenden oder Anhalten des Programms	`DoEvents, End, Exit, Stop`
Entscheidungen treffen	`Choose, If...Then...Else, Select Case, Switch`
Verwenden von Prozeduren	`Call, Function, Property Get, Property Let, Property Set, Sub`
Verzweigen	`GoSub...Return, GoTo, On Error, On...GoSub, On...GoTo`
• Compiler-Anweisung-Schlüsselwörter (Zusammenfassung)	
Definieren einer Compiler-Konstante	`#Const`
Kompilieren ausgewählter Code-Blöcke	`#If...Then...#Else`
• Datenfeld-Schlüsselwörter (Zusammenfassung)	
Ändern der Standarduntergrenze	`Option Base`
Deklarieren und Initialisieren eines Datenfeldes	`Dim, Private, Public, ReDim, Static`
Erneutes Initialisieren eines Datenfeldes	`Erase, ReDim`
Erstellen eines Datenfeldes	`Array`
Suchen der Grenzen eines Datenfeldes	`LBound, UBound`
Überprüfen eines Datenfeldes	`IsArray`

Weitere Informationen zu bestimmten Schlüsselwörtern finden Sie in Microsoft-VisualBasic- Hilfe.

Konstanten

Ein benanntes Element, das während der gesamten Programmausführung einen konstanten Wert behält. Eine Konstante kann ein Zeichenfolge- oder numerisches Literal, eine andere Konstante oder eine beliebige Kombination sein, die arithmetische oder logische Operatoren enthält. Jede Host-Anwendung kann ihren eigenen Satz von Konstanten bestimmen. Zusätzliche Konstanten können vom Benutzer mit der `Const`-Anweisung definiert werden. Sie können Konstanten überall in Ihrem Code anstelle der tatsächlichen Werte einsetzen.

Die folgenden Konstanten sind in Visual Basic für Applikationen definiert, um die Programmierung zu vereinfachen. Sie können daher überall im Code anstelle der tatsächlichen Werte verwenden:

- Compiler-Konstanten
- Datumskonstanten
- Dir-, GetAttr- und SetAttr-Konstanten
- Farbkonstanten
- IMEStatus-Konstanten
- Instr-, StrComp-Konstanten
- Kalender-Konstanten
- MsgBox-Konstanten
- QueryClose-Konstanten
- Shell-Konstanten
- StrConv-Konstanten
- Systemfarben-Konstanten
- Tasten-Code-Konstanten
- VarType-Konstanten
- Verschiedene Konstanten

MsgBox-Konstanten

Konstante	Wert	Beschreibung
`vbOK`	1	Schaltfläche OK gedrückt
`vbCancel`	2	Schaltfläche Abbrechen gedrückt
`vbAbort`	3	Schaltfläche Abbruch gedrückt
`vbRetry`	4	Schaltfläche Wiederholen gedrückt
`vbIgnore`	5	Schaltfläche Ignorieren gedrückt
`vbYes`	6	Schaltfläche Ja gedrückt
`vbNo`	7	Schaltfläche Nein gedrückt

Die Konstanten `vbOk`, `vbCancel`, `vbAbort`, `vbRetry`, `vbIgnore`, `vbYes` und `vbNo` sind beispielsweise Eigenschaften einer Schaltfläche und verweisen auf Rückgabe-Wert in `MsgBox`-Funktion. Mit der folgenden Anweisung können Sie prüfen, ob der Anwender die Yes- oder No-Schaltfläche gedrückt hat:

Beispiel 10)

```
Private Sub cmdloeschen_Click()
   If Me.cboPerson.Text = "" Then
     MsgBox "Sie haben keinen Eintrag gewählt"
     Beep
     Exit Sub
```

```
 Else
     Meldung = "Wollen Sie wirklich den Datensatz löschen?"
     Titel = "Datensatz löschen"
     ' Die Konstanten vbYesNo,  vbCritical und vbDefaultButton2?
     Stil = vbYesNo + vbCritical + vbDefaultButton2
     Ergebnis = MsgBox(Meldung, Stil, Titel)
     ' Die Konstante vbYes?
     If Ergebnis = vbYes Then
         Worksheets("tblZinsEin").Rows(varWelcherEintrag).Delete Shift:=x1up
         Maskeloeschen
         Einfuegen
     Else
         Exit Sub
     End If
  End If
End Sub
```


Weitere Informationen zu bestimmten Schlüsselwörtern finden Sie in Microsoft-VisualBasic- Hilfe.

Datentypen

Datentypen sind das, was Einsteigern bei anderen Programmiersprachen Kopfzerbrechen bereitet. Was es mit einem Datentyp auf sich hat? Für jede Variable oder Konstante wird ein bestimmter Platz im Speicher reserviert. Je nachdem, ob es sich um eine Zeichenkette oder eine Zahl handelt, wird ein mehr oder weniger großer Bereich des Speichers benötigt. Damit nun die Arbeit mit diesem Speicherbereich keine Fehler verursacht, muss der entsprechende Datentyp vorher deklariert werden.


Variable ist eine benannte Speicherposition, die Daten enthalten kann, die während der Programmausführung verändert werden können. Jede Variable hat einen Namen, der sie eindeutig in ihrem Gültigkeitsbereich identifiziert. Zusätzlich kann ein Datentyp angegeben werden. Variablennamen müssen mit einem Zeichen des Alphabets beginnen, innerhalb des Gültigkeitsbereichs müssen sie eindeutig sein, und sie dürfen nicht länger als 255 Zeichen sein. Variablennamen dürfen keinen Punkt und kein Typkennzeichen enthalten.

Es gibt prinzipiell diese Datentypen in VisualBais: `Byte`, `Boolean`, `Integer`, `Long`, `Currency`, `Decimal`, `Single`, `Double`, `Date`, `String`, `Object`, `Variant`. Dabei ist `Variant` (Voreinstellung) und die benutzerdefinierte Typen sind auch spezielle Objekttypen.

- `Boolean`-Datentyp: Variablen vom Datentyp `Boolean` werden als 16-Bit-Zahlen (2 Bytes) gespeichert, die nur die Werte `True` oder `False` annehmen können. Variablen vom Datentyp `Boolean` werden als Wahr oder Falsch ausgegeben, Beim Umwandeln von Werten des Datentyps `Boolean` in andere Datentypen wird False zu 0 und True zu -1.
- `Byte`-Datentyp: Variablen vom Datentyp `Byte` werden als einzelne 8-Bit-Zahlen (1 Byte) ohne Vorzeichen gespeichert und haben einen Wert im Bereich von 0 bis 255. Der Datentyp Byte ist zur Speicherung binärer Daten nützlich.
- `Currency`-Datentyp: Variablen vom Datentyp Currency werden als 64-Bit-Zahlen (8 Bytes) in einem ganzzahligen Format gespeichert und durch 10.000 dividiert, was eine Festkommazahl mit 15 Vorkomma- und 4 Nachkommastellen ergibt. Diese Darstellung

ergibt einen Wertebereich von -922.337.203.685.477,5808 bis 922.337.203.685.477,5807. Das Typkennzeichen für Currency ist das Zeichen (@). Der Datentyp Currency eignet sich besonders für Berechnungen mit Geldbeträgen und für Festkommaberechnungen, die eine hohe Genauigkeit erfordern.

- `Date`-Datentyp: Variablen vom Datentyp `Date` werden als 64-Bit-Gleitkommazahlen (8 Bytes) nach IEEE gespeichert und können ein Datum im Bereich vom 01. Januar 100 bis zum 31. Dezember 9999 und eine Uhrzeit im Bereich von 0:00:00 bis 23:59:59 speichern. Jeder gültige Wert eines Datums- oder Zeitliterals kann einer Variablen vom Datentyp `Date` zugewiesen werden. Ein Datumsliteral muss durch das Zeichen (#) eingeschlossen sein, zum Beispiel: #January 1, 1993# oder #1 Jan 93#.

Beispiel 11)

```
Sub test()
    Application.DisplayAlerts = False
    Heute = Now
    Verfalldatum = #1/15/1998# 'Die Datei ist veraltet
    'Verfalldatum = #15.02.98# 'Die Datei ist brauchbar
    If Verfalldatum > Heute Then
        MsgBox "Alles klar!"
    Else
        Prompt = "Diese Mappe wird geschlossen!"
        DialogArt = vbOK
        Title = "Arbeitsmappe zu alt"
        Antwort = MsgBox(Prompt, DialogArt, Title)
        Workbooks("datum").Close SaveChanges:=False
    End If
End Sub
```

- `Decimal`-Datentyp: Variablen des Datentyps `Decimal` werden als 96-Bit-Ganzzahlen (12 Bytes) ohne Vorzeichen mit einer variablen Potenz zur Basis 10 gespeichert. Die Potenz zur Basis 10 wird als Skalierungsfaktor verwendet und bestimmt die Anzahl der Nachkommastellen, die in einem Bereich von 0 bis 28 liegen kann. Der Datentyp `Decimal` kann nur mit einem Wert vom Typ `Variant` benutzt werden, d.h., Sie können keine Variable als Decimal deklarieren. Mit der `CDec`-Funktion können Sie jedoch einen Wert vom Typ `Variant` erstellen, dessen Untertyp `Decimal` ist.

- `Double`-Datentyp: Variablen vom Datentyp `Double` (Gleitkommazahl mit doppelter Genauigkeit) werden als 64-Bit-Gleitkommazahlen (8 Bytes) nach IEEE im Bereich von -1,79769313486232E308 bis -4,94065645841247E-324 für negative Werte und von 4,94065645841247E-324 bis 1,79769313486232E308 für positive Werte gespeichert. Das Typkennzeichen für `Double` ist das Zeichen (#).

- `Decimal`-Datentyp: Variablen vom Datentyp `Integer` werden als 16-Bit-Zahlen (2 Bytes) in einem Bereich von -32.768 bis 32.767 gespeichert. Das Typkennzeichen für `Integer` ist das Zeichen (%). Mit Variablen vom Datentyp `Integer` können Sie auch Aufzählungswerte darstellen. Ein Aufzählungswert besteht aus einer endlichen Menge eindeutiger ganzer Zahlen, von denen jede im verwendeten Kontext eine spezielle Bedeutung hat. Aufzählungswerte erlauben eine einfache Auswahl aus einer bestimmten Anzahl von Möglichkeiten, z.B. 0 = schwarz, 1 = weiß usw. Sie sollten die `Const`-Anweisung verwenden, um Konstanten für jeden Aufzählungswert zu definieren.

- `Long`-Datentyp: Variablen vom Datentyp `Long` (lange Ganzzahl) werden als 32-Bit-Zahlen (4 Bytes) mit Vorzeichen im Bereich von -2.147.483.648 bis 2.147.483.647 gespeichert. Das Typkennzeichen für `Long` ist das Zeichen (&).
- `Object`-Datentyp: Variablen vom Datentyp `Object` werden als 32-Bit-Adressen (4 Bytes) gespeichert, die auf Objekte in einer Anwendung verweisen. Einer Variablen, die als `Object` deklariert wurde, kann anschließend mit der `Set`-Anweisung ein Verweis auf jedes von der Anwendung erzeugte Objekt zugewiesen werden.
- `Single`-Datentyp: Variablen vom Datentyp `Single` (Gleitkommazahl mit einfacher Genauigkeit) werden als 32-Bit-Gleitkommazahlen (4 Bytes) nach IEEE im Bereich von -3,402823E38 bis -1,401298E-45 für negative Werte und von 1,401298E-45 bis 3,402823E38 für positive Werte gespeichert. Das Typkennzeichen für `Single` ist das Ausrufezeichen (!).
- `String`-Datentyp: Es gibt zwei Arten von Zeichenfolgen: Zeichenfolgen variabler Länge und Zeichenfolgen fester Länge.
  - ✓ Zeichenfolgen variabler Länge können bis zu 2 Milliarden (oder 2^31) Zeichen enthalten.
  - ✓ Zeichenfolgen fester Länge können 1 bis etwa 64 KB (2^16) Zeichen enthalten.

Die Codes für Zeichen vom Datentyp `String` liegen im Bereich von 0 bis 255 (einschließlich). Die ersten 128 Zeichen (0 bis 127) entsprechen den Buchstaben und Symbolen auf einer US-amerikanischen Standardtastatur. Diese ersten 128 Zeichen stimmen mit den im ASCII-Zeichensatz definierten Zeichen überein. Die zweiten 128 Zeichen (128 bis 255) sind Sonderzeichen, z.B. Buchstaben aus internationalen Alphabeten, Akzentzeichen, Währungssymbole und Symbole für mathematische Brüche. Das Typkennzeichen für `String` ist das Dollarzeichen ($).

- `Type`-Datentyp: Jeder Datentyp, den Sie mit der `Type`-Anweisung definieren. Benutzerdefinierte Datentypen können ein oder mehrere Elemente eines beliebigen Datentyps, eines Datenfeldes oder eines bereits bestehenden benutzerdefinierten Datentyps enthalten.

Beispiel 12)

```
Type Typ1
 Name1 As String         ' String-Variable für Namen.
 Geburtstag As Date      ' Date-Variable für Geburtstag.
 Geschlecht As Integer   ' Integer-Variable für
End Type                 '(0 für weiblich, 1 für männlich).PersonalNr = "5423";
```

- `Variant`-Datentyp: Der Datentyp `Variant` ist der Datentyp für alle Variablen, die nicht explizit (durch eine Anweisung wie `Dim`, `Private`, `Public` oder `Static`) als anderer Datentyp deklariert werden. Für den Datentyp `Variant` gibt es kein Typkennzeichen.
  - ✓ `Variant` ist ein besonderer Datentyp, der beliebige Daten mit Ausnahme von String-Daten fester Länge und benutzerdefinierten Typen enthalten kann. Ein `Variant` kann auch die speziellen Werte `Empty`, `Error`, `Nothing` und `Null` enthalten. Mit der Funktion `VarType` oder `TypeName` können Sie festlegen, wie die Daten in einer Variablen vom Datentyp `Variant` interpretiert werden.

- ✓ Als numerische Daten sind beliebige Ganzzahlen oder reelle Zahlen im Bereich von -1,797693134862315E308 bis -4,94066E-324 für negative Werte und von 4,94066E-324 bis 1,797693134862315E308 für positive Werte zulässig. Im Allgemeinen behalten numerische Daten vom Datentyp `Variant` den ursprünglich festgelegten Datentyp als Untertyp innerhalb des Variants bei. Wenn Sie zum Beispiel einem `Variant` einen Wert vom Datentyp `Integer` zuweisen, interpretieren alle nachfolgenden Operationen den Variant als Datentyp `Integer`. Wenn Sie jedoch mit einem `Variant` mit dem Typ `Byte`, `Integer`, `Long` oder `Single` eine arithmetische Operation ausführen und das Ergebnis den zulässigen Bereich für den ursprünglichen Datentyp überschreitet, wird das Ergebnis innerhalb des Variant automatisch zu dem nächstgrößeren Datentyp erweitert. `Byte` wird zu `Integer`, `Integer` wird zu `Long`, und `Long` bzw. `Single` werden zu `Double` umgewandelt. Werden die zulässigen Bereiche für den Datentyp `Currency`, `Decimal` oder `Double` in einem `Variant` überschritten, so tritt ein Fehler auf.
- ✓ Der Datentyp `Variant` kann anstelle jedes anderen Datentyps verwendet werden, um im Umgang mit Daten flexibler zu sein. Enthält eine `Variant`-Variable Ziffern, so können diese (je nach Kontext) entweder als Zeichenfolgendarstellung der Ziffern oder als deren tatsächlicher Wert interpretiert werden.

Beispiel 13)

```
Dim TestVar As Variant                ' Eine Variable als eine Zahl.
TestVar = 44137
Dim Matrix(1 To 3) As Variant         ' Ein Array mit 3 Elemente.
```


Weitere Informationen zu bestimmten Schlüsselwörtern finden Sie in Microsoft-VisualBasic- Hilfe.

Variablen

Variablen spielen eine grundsätzliche und wichtige Rolle in jeder Programmiersprache. Sie sind es im Grunde genommen, was unsere dynamische Software von heute vereinfacht. Eine Variable ist eigentlich ein Behälter für Informationen mit einem Wert. Der Behälter an sich bleibt dabei immer gleich, nur der Inhalt kann sich ändern. Jede Variable hat einen Namen, der sie eindeutig in ihrem Gültigkeitsbereich identifiziert. Zusätzlich kann ein Datentyp angegeben werden. Variablennamen müssen mit einem Zeichen des Alphabets beginnen, innerhalb des Gültigkeitsbereichs müssen sie eindeutig sein, und sie dürfen nicht länger als 255 Zeichen sein. Variablennamen dürfen keinen Punkt und kein Typkennzeichen enthalten.

Zur Deklaration von Variablen verwenden Sie normalerweise eine `Dim`-Anweisung. Eine Deklarationsanweisung kann innerhalb einer Prozedur zur Erstellung einer Variablen auf Prozedurebene oder zu Beginn eines Moduls im Deklarationsabschnitt zur Erstellung einer Variablen auf Modulebene plaziert werden.

Beispiel 12) In dem folgenden Beispiel wird die Variable strName erstellt und der Datentyp String angegeben.

```
Dim strName As String
```

Ist diese Anweisung Teil einer Prozedur, so kann die Variable strName nur in dieser Prozedur verwendet werden. Befindet sich diese Anweisung im Deklarationsabschnitt des Moduls, so ist die Variable strName für alle Prozeduren innerhalb des Moduls, aber nicht für Prozeduren in anderen Modulen des Projekts verfügbar. Damit diese Variable für alle Prozeduren des Projekts verfügbar ist, stellen Sie ihr die `Public`-Anweisung voran.

Beispiel 12)

```
Public strName As String
```


Weitere Informationen zur Benennung Ihrer Variablen finden Sie in der Visual Basic-Hilfe unter dem Thema "Namenskonventionen in Visual Basic".

Variablen können als einer der folgenden Datentypen deklariert werden: `Boolean`, `Byte`, `Integer`, `Long`, `Currency`, `Single`, `Double`, `Date`, `String` (für Zeichenfolgen variabler Länge), String * Länge (für Zeichenfolgen fester Länge), `Object` oder `Variant`. Wenn Sie keinen Datentyp angeben, wird der Datentyp `Variant` standardmäßig zugewiesen. Sie können auch einen benutzerdefinierten Typ über die `Type`-Anweisung erstellen.

Sie können mehrere Variable in einer Anweisung deklarieren. Wenn Sie einen Datentyp angeben möchten, müssen Sie den Datentyp für jede Variable angeben. In der folgenden Anweisung werden die Variablen intX, intY und intZ als `Integer` deklariert.

```
Dim intX As Integer, intY As Integer, intZ As Integer
```

In der folgenden Anweisung werden intX und intY als `Variant`, und nur intZ als `Integer` deklariert.

```
Dim intX, intY, intZ As Integer
```

Sie müssen die Datentypen der Variablen in der Deklarationsanweisung nicht angeben. Wenn Sie den Datentyp nicht angeben, so wird die Variable auf den Typ `Variant` festgelegt.

Sie können die `Public`-Anweisung zur Deklaration öffentlicher Variablen auf Modulebene verwenden.

```
Public strName As String
```

Öffentliche Variablen können in allen Prozeduren des Projekts verwendet werden. Wenn eine öffentliche Variable in einem Standardmodul oder einem Klassenmodul deklariert wird, kann diese auch in allen Projekten verwendet werden, die auf das Projekt verweisen, in dem die öffentliche Variable deklariert wurde.

Sie können die `Private`-Anweisung zur Deklaration privater Variablen auf Modulebene verwenden.

```
Private MeinName As String
```

Private Variablen können nur von Prozeduren des gleichen Moduls verwendet werden.


Wird die `Dim`-Anweisung auf Modulebene verwendet, gleicht sie der `Private`-Anweisung. Möglicherweise möchten Sie die `Private`-Anweisung verwenden, um die Lesbarkeit des Codes zu erhöhen und die Interpretation zu vereinfachen.

Wenn Sie die `Static`-Anweisung anstelle einer `Dim`-Anweisung verwenden, so behält die deklarierte Variable ihren Wert zwischen Aufrufen.

Bezeichnen einer Variablen

Der Name einer Variablen muss folgende Bedingungen erfüllen:

- Er muss ein Bezeichner sein.
- Er darf kein Schlüsselwort und kein Literal vom Typ `Boolean` (true oder false) sein.
- Er muss innerhalb seines Gültigkeitsbereichs eindeutig sein.

Gültigkeitsbereich einer Variablen

Der Gültigkeitsbereich ("Scope") einer Variablen bezieht sich auf den Bereich, in dem die Variable bekannt ist und in dem auf sie verwiesen werden kann.

Es gibt drei Gültigkeitsbereichebenen:

- Prozedurebene,
- private Modulebene
- öffentliche Modulebene.

Sie bestimmen den Gültigkeitsbereich einer Variablen bei deren Deklaration. Durch die explizite Deklaration aller Variablen können Fehler aufgrund von Namenskonflikten zwischen Variablen mit unterschiedlichen Gültigkeitsbereichen vermieden werden.

Definieren von Gültigkeitsbereichen auf Prozedurebene

Eine innerhalb einer Prozedur definierte Variable oder Konstante ist außerhalb dieser Prozedur nicht verfügbar. Nur die Prozedur, die die Variablendeklaration enthält, kann diese Variable verwenden. Im folgenden Beispiel zeigt die erste Prozedur ein Meldungsfeld mit einer Zeichenfolge an. Die zweite Prozedur zeigt ein leeres Meldungsfeld an, da die Variable strMldg für die erste Prozedur lokal ist.

Beispiel 13)

```
Sub LokaleVariable()
   Dim strMldg As String
   strMldg = "Diese Variable kann nur innerhalb " &  "dieser Prozedur verwendet werden."
   MsgBox strMldg
End Sub

Sub AußerhalbGültigkeitsbereich()
   MsgBox strMldg
End Sub
```

Definieren von Gültigkeitsbereichen auf privater Modulebene

Sie können Variablen und Konstanten auf Modulebene im Deklarationsabschnitt eines Moduls definieren. Variablen auf Modulebene können entweder öffentlich oder privat sein. Öffentliche Variablen sind für alle Prozeduren in allen Modulen eines Projekts verfügbar, private Variablen sind nur für die Prozeduren in diesem Modul verfügbar. Standardmäßig werden Variablen mit der `Dim`-Anweisung im Deklarationsabschnitt im privaten Gültigkeitsbereich deklariert. Durch Voranstellen des Schlüsselworts `Private` jedoch wird der Gültigkeitsbereich der Variablen in Ihrem Code erkennbar.

Im folgenden Beispiel ist die Zeichenfolgenvariable strMldg für alle in dem Modul definierten Prozeduren verfügbar. Bei Aufruf der zweiten Prozedur werden die Inhalte der Zeichenfolgenvariablen strMldg in einem Dialogfeld angezeigt.

Beispiel 14)

```
' Fügen Sie zum Deklarationsabschnitt des Moduls folgenden Code hinzu.
Private strMldg As String
Sub InitialisierePrivateVariable()
  strMldg = "Diese Variable kann nur innerhalb " &  "dieses Moduls verwendet werden."
End Sub

Sub VerwendePrivateVariable()
   MsgBox strMldg
End Sub
```

Definieren von Gültigkeitsbereichen auf öffentlicher Modulebene

Wenn Sie eine Variable auf Modulebene als öffentlich deklarieren, ist sie für alle Prozeduren in diesem Projekt verfügbar. Im folgenden Beispiel kann die Zeichenfolgenvariable strMldg von jeder Prozedur in jedem Modul dieses Projekts verwendet werden.

```
' Fügen Sie diesen Code in den Deklarationsabschnitt des Moduls ein.
Public strMldg As String
```

Alle Prozeduren (ausgenommen Ereignisprodezuren) sind standardmäßig öffentlich. Wenn Visual Basic eine Ereignisprozedur erstellt, wird das Schlüsselwort `Private` automatisch vor der Prozedurdeklaration eingefügt. Für alle anderen Prozeduren müssen Sie die Prozedur explizit mit dem Schlüsselwort Private deklarieren, wenn die Prozedur nicht öffentlich sein soll.

Typumwandlungen

In jeder Sprache kommt vor, dass der Datentyp einer Variablen innerhalb ihres Gültigkeitsbereiches geändert werden muss. In VisualBasic legt jede Funktion für einen bestimmten Datentyp zwingend einen Ausdruck fest. Die folgenden Funktionen werden in VisualBasic für Typumwandlungen verwendet:

Funktion	Rückgabetyp	Bereich des Arguments Ausdruck
CBool	Boolean	Eine gültige Zeichenfolge oder ein gültiger numerischer Ausdruck.
CByte	Byte	0 bis 255.
CCur	Currency	-922.337.203.685.477,5808 bis 922.337.203.685.477,5807.
CDate	Date	Ein beliebiger gültiger Datumsausdruck.
CDbl	Double	-1,79769313486232E308 bis -4,94065645841247E-324 für negative Werte; 4,94065645841247E-324 bis 1,79769313486232E308 für positive Werte.
CDec	Decimal	+/-79.228.162.514.264.337.593.543.950.335 für skalierte Ganzzahlen, d.h. Zahlen ohne Dezimalstellen. Für Zahlen mit 28 Dezimalstellen gilt der Bereich+/-7,9228162514264337593543950335. Die kleinste mögliche Zahl ungleich Null ist 0,0000000000000000000000000001.
CInt	Integer	-32.768 bis 32.767; Nachkommastellen werden gerundet.
CLng	Long	-2.147.483.648 bis 2.147.483.647; Nachkommastellen werden gerundet.
CSng	Single	-3,402823E38 bis -1,401298E-45 für negative Werte; 1,401298E-45 bis 3,402823E38 für positive Werte.
Cvar	Variant	Numerische Werte im Bereich des Typs Double. Nichtnumerische Werte im Bereich des Typs String.

CStr	String	Rückgabe für CStr hängt vom Argument Ausdruck ab.

Das erforderliche Argument Ausdruck ist ein Zeichenfolgenausdruck oder ein numerischer Ausdruck.

In diesem Beispiel wird die `CStr`-Funktion verwendet, um einen numerischen Wert in den Typ `String` umzuwandeln.

Beispiel 15)

```
Dim TestDouble, TestString
TestDouble = 437.324            ' TestDouble hat den Typ Double.
TestString = CStr(TestDouble)   ' TestString enthält "437,324".
```

In diesem Beispiel wird die `CDate`-Funktion verwendet, um eine Zeichenfolge in einen Wert vom Typ `Date` umzuwandeln. In der Regel sollten Sie Datums- und Zeitangaben nicht als Zeichenfolgen direkt in den Code eingeben, wie in diesem Beispiel. Verwenden Sie stattdessen Datums- und Zeitliterale, wie #2/12/1969#, #4:45:23 PM#.

Beispiel 16)

```
Dim TestDatum, TestKurzesDatum, TestZeit, TestKurzeZeit
TestDatum = "12. Februar 1969"       ' Datum definieren.
TestKurzesDatum = CDate(TestDatum)   ' In Datentyp Date umwandeln.
TestZeit = "16:35:47"                ' Zeit definieren.
TestKurzeZeit = CDate(TestZeit)      ' In Datentyp Date umwandeln.
```

Arrays

Arrays stellen eine besondere Art von Variablen dar. Im Gegensatz zu den üblichen Variablen sind sie in der Lage, mehrere Werte aufzunehmen, wie es im obigen Abschnitt erklärt wurde. Dabei werden wieder zwei Arten von Arrays unterschieden. Zum einen können Arrays werte in Form einer einfachen Liste oder auch Aufzählung aufnehmen. Hier ist die Rede von *eindimensionalen Arrays*. Zum anderen können Arrays auch Werte in Form einer Tabelle aufnehmen, hier ist die Rede von *zweidimensionalen Arrays*.

Eindimensionale Arrays

Eindimensionale Arrays nehmen Werte in der Art einer Liste auf. Um auf die Werte innerhalb dieser Liste zuzugreifen, werden diese mit einem fortlaufenden Index versehen. In VisualBasic gibt die `Array`-Funktion, mit der ein Wert vom Typ `Variant` zurückgibt. Dieser Wert enthält ein Datenfeld.

Syntax

```
Array(ArgListe)
```

Das erforderliche Argument ArgListe enthält eine durch Kommas getrennte Liste von Werten, die den in `Variant` enthaltenen Elementen des Datenfeldes zugewiesen werden. Werden keine Argumente angegeben, so erstellt die Funktion ein Null-Datenfeld.

Der Zugriff auf ein beliebiges Element eines Datenfeldes erfolgt durch Angabe des Variablennamens mit einem nachfolgenden Klammernpaar, das die Indexnummer des gewünschten Elements enthält. Im folgenden Beispiel erstellt die erste Anweisung die Variable DayOfWeek als `Variant`. Die zweite Anweisung weist der Variablen DayOfWeek ein

Datenfeld zu. Die letzte Anweisung weist den Wert, der im zweiten Datenfeldelement enthalten ist, einer anderen Variablen zu.

Beispiel 17)

```
Dim DayOfWeek As Variant
DayOfWeek = Array("Mo", "Di", "Mi", "Do", "Fr", "Sa", "So");
B = DayOfWeek(2)                ' Also zweites Element ist Di
```

Die untere Grenze für ein mit der `Array`-Funktion erstelltes Datenfeld ist immer Null. Im Gegensatz zu anderen Datenfeldtypen ist es nicht von der unteren Grenze betroffen, die mit der Option `Base`-Anweisung festgelegt wurde.

Ein `Variant`, der nicht als Datenfeld deklariert ist, kann trotzdem ein Datenfeld enthalten. Eine Variable vom Typ `Variant` kann ein Datenfeld eines beliebigen Typs enthalten, außer Zeichenfolgen konstanter Länge und benutzerdefinierte Typen. Obwohl ein `Variant`, der ein Datenfeld enthält, sich konzeptionell von einem Datenfeld unterscheidet, dessen Elemente vom Typ `Variant` sind, erfolgt der Zugriff auf die Datenfeldelemente in gleicher Weise.

Datenfeldern

Sie können ein Datenfeld deklarieren, das mit einer Gruppe von Werten des gleichen Datentyps arbeitet. Ein Datenfeld ist eine einzelne Variable mit mehreren Feldern, in denen Werte gespeichert werden, während eine typische Variable nur über ein Speicherfeld verfügt, in dem nur ein Wert gespeichert werden kann. Sie können auf das Datenfeld als Ganzes verweisen, wenn Sie auf alle darin befindlichen Werte verweisen möchten, oder auf die einzelnen Elemente.

Damit Sie z.B. die täglichen Kosten für jeden Tag des Jahres speichern können, deklarieren Sie eine Datenfeldvariable mit 365 Elementen anstelle von 365 Variablen. Jedes Element in einem Datenfeld enthält einen Wert. Die folgende Anweisung deklariert die Datenfeldvariable curKosten mit 365 Elementen. Standardmäßig wird ein Datenfeld beginnend mit Null (0) indiziert, so daß die obere Grenze des Datenfeldes statt 365 nur 364 entspricht.

```
Dim curKosten (364) As Currency
```

Damit Sie den Wert eines einzelnen Elements festlegen können, geben Sie den Index des Elements an. Das folgende Beispiel weist jedem Element im Datenfeld den Anfangswert 20 zu.

```
Sub FülleDatenfeld ()
   Dim curKosten (364) As Currency
   Dim intI As Integer
   For intI = 0 to 364
         curKosten (intI) = 20
   Next
End Sub
```

Sie können die Option `Base`-Anweisung am Anfang eines Moduls dazu verwenden, den Standardindex des ersten Elements von 0 auf 1 zu ändern. Im folgenden Beispiel ändert die Option `Base`-Anweisung den Index für das erste Element und die `Dim`-Anweisung deklariert die Datenfeldvariable curKosten mit 365 Elementen.

```
Option Base 1
Dim curKosten(365) As Currency
```

Sie können auch explizit die untere Grenze eines Datenfelds über den To-Abschnitt festlegen.

Beispiel 18)

```
Dim curKosten(1 To 365) As Currency
Dim strWochentag(7 To 13) As String
```

Es gibt zwei Möglichkeiten, Datenfelder für Werte des Typs Variant zu erstellen. Sie können ein Datenfeld als Datentyp `Variant` deklarieren. Beispiel:

```
Dim varDaten(3) As Variant
varDaten(0) = "Christine Müller"
varDaten(1) = "44137 Dortmund"
varDaten(2) = 38
varDaten(3) = Format("06-09-1952", "General Date")
```

Sie können aber auch das von der `Array`-Funktion zurückgegebene Datenfeld einer `Variant`-Variablen zuweisen.

Beispiel 19)

```
Dim varDaten As Variant
varDaten = Array("Christine Müller", "44137 Dortmund", 38, _
Format("06-09-1952", "Allgemeines Datum"))
```

Sie kennzeichnen die Elemente in einem Datenfeld von `Variant`-Werten über den Index, unabhängig von der bei der Erstellung des Datenfeldes verwendeten Methode. Die folgende Anweisung kann z.B. zu jedem der vorhergehenden Beispiele hinzugefügt werden.

```
MsgBox "Daten für " & varDaten(0) & " wurden aufgezeichnet."
```

mehrdimensionale Datenfelder (mehrdimensionale Arrays)

In Visual Basic können Sie Datenfelder mit bis zu 60 Dimensionen deklarieren. Die folgende Anweisung deklariert z.B. ein zweidimensionales, 5-zu-10-Datenfeld.

```
Dim sngMulti(1 To 5, 1 To 10) As Single
```

Wenn Sie sich das Datenfeld als Matrix vorstellen, so stellt das erste Argument die Zeilen und das zweite Argument die Spalten dar.

Verwenden Sie verschachtelte `For...Next`-Anweisungen für die Verarbeitung mehrdimensionaler Datenfelder. Die folgende Prozedur füllt ein zweidimensionales Datenfeld mit `Single`-Werten.

```
Sub FülleDatenfeldMulti ()
   Dim intI As Integer, intJ As Integer
   Dim sngMulti(1 To 5, 1 To 10) As Single
   ' Ausfüllen des Datenfeldes mit Werten.
   For intI = 1 To 5
      For intJ = 1 To 10
           sngMulti(intI, intJ) = intI * intJ
           Debug.Print sngMulti(intI, intJ)
      Next intJ
   Next intI
End Sub
```

Weitere Informationen zu bestimmten Schlüsselwörtern finden Sie in Microsoft-VisualBasic- Hilfe.

Operatoren

Operatoren werden öfter benötigt als man im ersten Moment glauben mag. Operatoren sind Zeichen, die festlegen, wie Werte in einem Ausdruck kombiniert, verglichen oder geändert werden. Die Elemente, auf die der Operator angewendet wird, werden als Operanden bezeichnet. Operatoren werden in verschiedene Gruppe eingeteilt, wobei jede dieser Gruppen eigene Aufgaben zu erfüllen hat.

Arithmetische Operatoren

Arithmetische Operatoren addieren, subtrahieren, multiplizieren, dividieren und führen weitere arithmetische Operationen durch. In der folgenden Tabelle sind die numerischen Operatoren in ActionScript aufgeführt:

Operator	Bezeichnung	Beispiel	Beschreibung
+	Addition	`Res = a + b`	Summe von a und b
-	Subtraktion	`Res = a - b`	Differenz von a und b
*	Multiplikation	`Res = a * d`	Produkt von a und d
/	Division	`Res = a / b`	Dividieren a durch b , Res ist Fließkommazahl
\	Division	`Res = d / a;`	Dividieren a durch b , Res ist Ganzzahl
Mod	Modulo	`Res = a Mod b`	Rest der Division von a und b
^	Potenzieren	`Res = a^b`	Potenzieren von a mit Exponent b

Vergleichsoperatoren

Diese Operatoren gehören zu den mit Sicherheit am meisten benötigten. Sie vergleichen Werte von Ausdrücken und geben einen Wert vom Typ `Boolean` (true oder false) zurück. Diese Operatoren werden meist in Schleifen und bedingten Anweisungen verwendet. In diesem Beispiel werden verschiedene Einsatzmöglichkeiten für Vergleichsoperatoren zum Vergleichen von Ausdrücken aufgezeigt.

Beispiel 20)

```
Dim Ergebnis, Var1, Var2
Ergebnis = (45 < 35)                ' Liefert False.
Ergebnis = (45 = 45)                ' Liefert True.
Ergebnis = (4 <> 3)                 ' Liefert True.
Ergebnis = ("5" > "4")              ' Liefert True.
Var1 = "5": Var2 = 4                ' Variablen initialisieren.
Ergebnis = (Var1 > Var2)            ' Liefert True.
Var1 = 5: Var2 = Empty
Ergebnis = (Var1 > Var2)            ' Liefert True.
Var1 = 0: Var2 = Empty
Ergebnis = (Var1 = Var2)            ' Liefert True.
```

In der folgenden Tabelle sind die Vergleichsoperatoren in VisualBasic aufgeführt:

Operator	Bezeichnung	Beispiel	Beschreibung
<	Kleiner als	If(a < b)	a ist kleiner als b.
>	Größer als	If(b > a)	b ist größer als b.
<=	Kleiner oder gleich	If(a <= b)	a ist kleiner oder gleich b.
>=	Größer oder gleich	If(a >= b)	a ist kleiner oder gleich b.
=	gleich	If(a = b)	a ist gleich b.
<>	ungleich	If(a <> b)	a ist kleiner oder größer als b.

Logische Operatoren

Logische Operatoren vergleichen Werte vom Typ Boolean (true und false) und geben einen dritten Wert vom Typ Boolean zurück. Wenn beide Operanden z.B True ergeben, gibt der logische Operator UND (&&) das Ergebnis true zurück. Wenn einer oder beide Operanden true ergeben, gibt der logische Operator ODER (||) das Ergebnis true zurück. Logische Operatoren werden häufig in Verbindung mit Vergleichsoperatoren verwendet, um die Bedingung für eine if-Aktion festzulegen. In diesem Beispiel wird der Operator And verwendet, um zwei Ausdrücke mit einer logischen Konjunktion zu verknüpfen.

Beispiel 21)

```
Dim A, B, C, D, Test1
A = 10: B = 8: C = 6: D = Null        ' Variablen initialisieren.
Test1 = A > B And B > C               ' Liefert True.
Test1 = B > A And B > C               ' Liefert False.
Test1 = A > B And B > D               ' Liefert Null.
Test1 = A And B                       ' Liefert 8 (Bit-weiser Vergleich).
```

In der folgenden Tabelle sind die logischen Operatoren in VisualBasic aufgeführt:

Operator	Bezeichnung	Beispiel	Beschreibung
And	Logisches UND	a1 And a2	Ergibt true, wenn beide a1 und a2 true sind.
Or	Logisches ODER	a1 Or a2	Ergibt true, wenn mindestens einer a1 oder a2 true sind.
Eqv	Logisches Äquivalenz	a1 Eqv a2	Ergibt true, wenn entweder a1 und a2 true oder false sind.
Not	logische Negation	Not a1	Ergibt true, wenn a1 false ist.
Xor	logischen Exklusion	a1 Xor a2	Ergibt true, wenn entweder a1 oder a2 true ist, aber nicht beides

Verkettungsoperatoren

Verkettungsoperatoren dienen zu Verkettung zweier Zeichenketten (in diesem Fall mit dem Operator &) und Addition von zweier Zahlen (in diesem Fall mit dem Operator +).

Wenn Sie den Operator + verwenden, können Sie nicht immer bestimmen, ob eine Addition oder eine Zeichenverkettung erfolgt. Für die Verkettung zweier Zeichenfolgen sollten Sie den Operator & verwenden, um Mehrdeutigkeiten auszuschließen und Code zu erstellen, der sich selbst dokumentiert.

Operator	Bezeichnung	Beispiel	Beschreibung
&	Bitweises UND	`a & b`	a und b werden verkettet.
+	Addition zweier zahlen	`Res = a + b`	Es werden alle Bits beeinflusst, die in a oder b gesetzt sind.

In diesem Beispiel wird der Operator + verwendet, um Zahlen zu addieren. Der Operator + kann auch Zeichenfolgen verketten. Um Mehrdeutigkeiten zu vermeiden, sollten Sie jedoch dazu den Operator & verwenden. Wenn die Komponenten eines Ausdrucks, der mit dem Operator + erstellt wurde, sowohl Zeichenfolgen als auch Zahlen umfassen, wird das arithmetische Ergebnis zugewiesen. Wenn die Komponenten ausschließlich Zeichenfolgen sind, werden diese Zeichenfolgen verkettet.

Beispiel 22)

```
Dim hufPlusmwzhu, huf, mwzhu
huf = 38, mwzhu=2
HufPlusmwzhu = huf + mwzhu        ' Liefert 40
```

In diesem Beispiel wird der Operator & verwendet, um Zeichenfolgen zu verketten.

Beispiel 23)

```
Dim Textkmg
Textkmg = "Körpermaße" & " Größe"    ' Liefert "Körpermaße Größe"
```

Operatorvorrang

Wenn ein Ausdruck mehrere Operationen enthält, werden die einzelnen Teilausdrücke in einer bestimmten Rangfolge ausgewertet und aufgelöst, die als Operatorvorrang bezeichnet wird. Wenn Ausdrücke Operatoren aus mehreren Kategorien enthalten, werden zunächst die **arithmetischen Operatoren**, dann die **Vergleichsoperatoren** und zuletzt die **logischen Operatoren** ausgewertet. Die Vergleichsoperatoren haben alle dieselbe Priorität und werden daher von links nach rechts in der Reihenfolge ihres Auftretens ausgewertet. Für die arithmetischen und logischen Operatoren gilt die folgende Rangfolge:

Arithmetisch	Vergleich	Logisch
Potenzierung (^)	Gleich (=)	`Not`
Negation (-)	Ungleich (<>)	`And`
Multiplikation und Division (*, /)	Kleiner als (<)	`Or`
Ganzzahldivision (\)	Größer als (>)	`Xor`
Restwert (Mod)	Kleiner oder gleich (<=)	`Eqv`

Addition und Subtraktion (+, –)	Größer oder gleich (>=)	Imp
Zeichenverkettung (&)	Like, Is	

Multiplikation und Division innerhalb eines Ausdrucks sind gleichrangig und werden von links nach rechts in der Reihenfolge ihres Auftretens ausgewertet. Dasselbe gilt für Additionen und Subtraktionen, die zusammen in einem Ausdruck auftreten. Mit Klammern kann diese Rangfolge außer Kraft gesetzt werden, damit bestimmte Teilausdrücke vor anderen Teilausdrücken ausgewertet werden. In Klammern gesetzte Operationen haben grundsätzlich Vorrang. Innerhalb der Klammern gilt jedoch wieder die normale Rangfolge der Operatoren. Der Zeichenverkettungsoperator (&) ist zwar kein arithmetischer Operator, liegt aber in der Rangfolge zwischen den arithmetischen Operatoren und den Vergleichsoperatoren. Der Operator `Like` ist eigentlich ein Operator zum Mustervergleich, wird aber in der Rangfolge den anderen Vergleichsoperatoren gleichgestellt. Der Operator `Is` dient zum Vergleichen von Verweisen auf Objekte. Er vergleicht nicht die Objekte oder deren Werte, sondern überprüft lediglich, ob sich zwei Objektverweise auf dasselbe Objekt beziehen.

 Weitere Informationen Operatoren finden Sie in der Visual Basic-Hilfe unter dem Thema "Operatoren in Visual Basic".

Kontrollstrukturen

Um Variablen und Objekten manipulieren und steuern zu können oder den Zustand des Programms zu überprüfen, kommen oft Anweisungen und Schleifen zum Einsatz. Bei solchen Anweisungen und Schleifen handelt es sich um die Auswertung und Überprüfung Zuständen sowie Werten. Zu Bewältigung solcher Aufgaben können so genannte Kontrollstrukturen eingesetzt werden. Das Prinzip solcher Kontrollstrukturen ist immer gleich: Es findet ein Vergleich oder eine Überprüfung statt, deren Ergebnis entweder den Wert als `Boolean` true (wahr) oder false (falsch) hat. Je nachdem, wie die Überprüfung oder Vergleich ausfällt, kann das Programm in eine bestimmte Richtung verzweigt und weitergeführt werden. Allerdings unterscheiden sich die Kontrollstrukturen wesentlich in ihrem Aufbau und ihrer Funktionsweise.

Fallunterscheidungen

`if...Then...Else`-Anweisung

Die `if...Then...Else`-Anweisung führt eine Gruppe von Anweisungen aus, wenn bestimmte Bedingungen erfüllt sind, die vom Wert eines Ausdrucks abhängen.

Syntax:

```
If Bedingung Then
   [Anweisungen]
[ElseIf Bedingung-n Then
   [elseifAnweisungen] ...
[Else
   [elseAnweisungen]]
End If
```

Beispiel 24) In diesem Beispiel wird die Blockform und die einzeilige Form der `If...Then...Else`-Anweisung demonstriert. Eine Zahl in der Zelle A1 wird überprüft. Das Ergebnis wird in der Zelle A2 angezeigt.

```
Dim Zahl As Integer
Dim Tabelle As Object
Set Tabelle = WorkSheets("Tabelle1")
Zahl = Tabelle.Cells(1,1).Value  'Variable initialisieren.
If Zahl < 10 Then
    Tabelle.Cells(2,1).Value= "Zahl ist kleiner als 10."
ElseIf Zahl < 100 Then
 'Bedingung ist erfüllt, also wird die nächste Anweisung ausgeführt.
    Tabelle.Cells(2,1).Value= "Zahl ist kleiner als 100."
Else
    Tabelle.Cells(2,1).Value= "Diese Zahl ist nicht gültig."
End If
```

`Select Case`-Anweisung

Bei `IF`-Konstruktionen lassen sich immer nur zwei alternative Programmteile ausführen, da eine Bedingung als logischer Ausdruck nur die Werte True bzw. False ergeben kann. Bei einer mehrseitigen Auswahl testen Sie den Wert einer Variablen oder eines komplexen Ausdrucks. Diese Variable bzw. dieser Ausdruck wird Testausdruck bezeichnet. In Abhängigkeit vom Wert des Testausdrucks können je nach Fall (`case`) verschiedene Anweisungesblöcke ausgeführt werden. Die mehrseitige Auswahl wird daher auch Testausdruck(Selection) genannt. Also `Select case`-Anweisung führt eine von mehreren Gruppen von Anweisungen aus, abhängig vom Wert eines Ausdrucks.

Syntax:

```
Select Case Testausdruck
  [Case Ausdrucksliste-n
     [Anweisungen-n]] ...
  [Case Else
     [elseAnw]]

End Select
```

Beispiel 25) In diesem Beispiel wird die **`Select Case`**-Anweisung verwendet, um den Wert einer Variablen auszuwerten. Der zweite **`Case`**-Abschnitt enthält den Wert der ausgewerteten Variablen, und nur die zugehörige Anweisung wird ausgeführt.

```
Dim Zahl
Zahl = 8    ' Variable initialisieren.
Select Case Zahl    ' Zahl auswerten.
Case 1 To 5    ' Zahl von 1 bis 5.
    Debug.Print "Zahl von 1 bis 5"
' Das ist der einzige Case-Abschnitt, der True ergibt.
Case 6, 7, 8    ' Zahl von 6 bis 8.
    Debug.Print "Zahl von 6 bis 8"
```

```
Case Is 9 To 10     ' Zahl ist 9 oder 10.
Debug.Print "Größer als 8"
Case Else     ' Andere Werte.
    Debug.Print "Außerhalb von 1 bis 10"
End Select
```

Weitere Informationen über die Fallunterscheidungen wie Select Case-Anweisung, Switch-Funktion, `#if...#Then...#Else`-Anweisung finden Sie in der Visual Basic-Hilfe unter dem Thema "Schlüsselwörter in Visual Basic".

Schleifen

Eine weitere Form der Kontrollstrukturen sind Schleifen. Während eine Fallunterscheidung nur ein einziges Mal durchlaufen wird, sind bei Schleifen mehrere Durchläufe, also Wiederholungen möglich. Jeder Schleifentypus weist einen Schleifenkopf auf, der durch eine Bedingung gekennzeichnet ist. Diese Bedingung legt die Anzahl der Aufrufe des Anweisungsblocks fest. Wenn die Bedingung einer Schleife nicht erfüllt ist, wird die Schleife ignoriert und die Programmausführung nach dem Funktionsblock der Schleife fortgesetzt. Falls ein vorzeitiges Abbrechen der Iteration durch den Funktionsblock einer Schleife erwünscht ist, so kann dies durch einen Aufruf der Funktion `Exit` erzielt werden.

`For Each...Next`-Anweisungen

Die `for Each...Next`-Schleife findet ihren Einsatz in Bereichen, in denen es auf die Ausführung einer Anzahl von Anweisungen in einer bestimmten Häufigkeit ankommt. Sie wiederholen einen Block von Anweisungen für jedes Objekt in einer Auflistung oder jedes Element in einem Datenfeld.

Syntax:

```
For Each Element In Gruppe
   [Anweisungen]
   [Exit For]
   [Anweisungen]
Next [Element]
```

Visual Basic stellt automatisch bei jedem Schleifendurchlauf eine Variable ein. Die folgende Prozedur z.B. schließt alle Formulare mit Ausnahme des Formulars, das die auszuführende Prozedur enthält.

```
Sub SchließeFormulare()
      For Each frm In Application.Forms
            If frm.Caption <> Screen. ActiveForm.Caption Then frm.Close
      Next
End Sub
```

In diesem Beispiel wird über die Zellen A1:D10 in Tabelle1 eine Schleife durchlaufen. Wenn eine der Zellen einen Wert unter 0,001 besitzt, ersetzt der Code den Wert mit 0 (Null).

```
For Each zelle In Worksheets("Tabelle1").Range("A1:D10")
      If zelle.Value < .001 Then
            zelle.Value = 0
      End If
Next zelle
```

In diesem Beispiel wird die **`For Each...Next`**-Anweisung verwendet, um die **`Text`**-Eigenschaft aller Elemente in einer Auflistung auf das Vorhandensein der Zeichenfolge "Hallo" zu untersuchen. Im Beispiel ist Objekt1 ein Objekt, das Text enthalten kann und ein Element der Auflistung Auflistung1 darstellt. Die beiden Bezeichnungen dienen lediglich Demonstrationszwecken.

```
Dim Gefunden, Objekt1, Auflistung1
Gefunden = False    ' Variable initialisieren.
For Each Objekt1 In Auflistung1    ' Alle Elemente durchlaufen.
    If Objekt1.Text = "Hallo" Then    ' Wenn Text gleich "Hallo",
        Gefunden = True    ' Gefunden auf True setzen.
        Exit For    ' Schleife verlassen.
    End If
Next
```

`For...Next`-Anweisungen

Sie können `For...Next`-Anweisungen verwenden, um einen Block von Anweisungen eine unbestimmte Anzahl von Wiederholungen ausführen zu lassen. For-Schleifen verwenden eine Zählervariable, deren Wert mit jedem Schleifendurchlauf erhöht oder verringert wird.

Syntax:

```
For Zähler = Anfang To Ende [Step Schritt]
  [Anweisungen]
  [Exit For]
  [Anweisungen]
Next [Zähler]
```

Innerhalb einer Schleife kann eine beliebige Anzahl von `Exit For`-Anweisungen an beliebiger Stelle als alternative Möglichkeit zum Verlassen der Schleife verwendet werden. `Exit For` wird oft in Zusammenhang mit der Auswertung einer Bedingung (zum Beispiel `If...Then`) eingesetzt und überträgt die Steuerung an die unmittelbar auf Next folgende Anweisung. (Siehe Beispiel 4)

`Do...Loop`-Anweisung

Wiederholt einen Block mit Anweisungen, solange eine Bedingung den Wert True hat oder bis eine Bedingung den Wert True erhält.

Syntax:

```
Do [{While | Until} Bedingung]
  [Anweisungen]
  [Exit Do]
  [Anweisungen]
Loop
```

Innerhalb einer Do…Loop-Anweisung kann eine beliebige Anzahl von Exit Do-Anweisungen an beliebiger Stelle als Alternative zum Verlassen einer Do…Loop-Anweisung verwendet werden. Exit Do wird oft in Zusammenhang mit der Auswertung einer Bedingung (zum Beispiel If...Then) eingesetzt und hat zur Folge, dass die Ausführung mit der ersten Anweisung im Anschluss an Loop fortgesetzt wird.
Wir formen das Beispiel 4 in diesem Abschnitt in einer Do…Loop –Schleife um.

Beispiel 26)

```
Sub Einfuegen()
   Dim i As Integer, flag As Boolean
   i = 1
   flag = True
   Do While flag
      If Worksheets("tblZinsEIN").Cells(i, 1).Value > "" Then
         frmZins.cboPerson.AddItem Worksheets("tblBSTEIN").Cells(i, 1).Value
         i = i + 1
      Else
         flag = False
      End If
   Loop
End Sub
```

In dem Beispiel werden die Daten von der ersten Spalte des Tabellenblattes tblZinsEIN in das Kombinationsfeld cboPerson, das sich im Dialogblatt frmZins befindet, eingefügt. Diese Schleife wird erst verlassen, wenn die Bedingung false wird. D.h. der Inhalt der Zelle ist leer.

With-Anweisung

In Visual Basic gehört die With-Anweisung zu den Schleifen, weil Sie damit Sie eine Reihe von Anweisungen für ein bestimmtes Objekt ausführen können, ohne den Namen des Objekts mehrmals angeben zu müssen. Wenn Sie zum Beispiel mehrere Eigenschaften eines bestimmten Objekts verändern möchten, sollten Sie die Zuweisungsanweisungen für die Eigenschaft in der With-Kontrollstruktur unterbringen. Sie brauchen dann den Namen des Objekts nicht bei jeder einzelnen Zuweisung, sondern nur einmal zu Beginn der Kontrollstruktur anzugeben.

Syntax:

```
With Objekt
   [Anweisungen]
End With
```

Das folgende Beispiel veranschaulicht die Verwendung der With–Anweisung, um mehreren Eigenschaften desselben Objekts Werte zuzuweisen.

```
With WorkSheets("tblZinsAUS").Range("A1")
      .Height = 20
      .Width = 50
      .Caption = "Schönen Tag noch"
End With
```

Funktionen

Eine Funktion besteht aus einem geschlossenen Codeblock, der unter einem bestimmten Namen aufgerufen und ausgeführt werden kann. Dabei spricht man von einer *Funktion*. Eine Funktion kann eine beliebige Anzahl von ausführbaren Anweisungen enthalten. Außerdem ist es möglich, aus einer Funktion heraus weitere Funktionen aufzurufen. Die Funktion kann sich an einer beliebigen Stelle innerhalb des Codes befinden und auch von jeder beliebigen Stelle aus aufgerufen werden. Nach dem Beenden der Funktion wird der Programmablauf an der Stelle fortgesetzt, an der er durch den Aufruf der Funktion unterbrochen wurde.

Vorteile von Funktionen

Mit Hilfe von Funktionen können Sie eigene, in sich abgeschlossene Visual Basic-Prozeduren programmieren, die Sie dann über den Aufruf der Funktion ausführen können. Dabei können Sie bestimmen, bei welchem Ereignis (zum Beispiel, wenn der Anwender einen Button anklickt) die Funktion aufgerufen und ihr Programmcode ausgeführt wird. So vermeiden Sie mehrfache Nutzung verschiedener Programmteile. Mit Hilfe einer Funktion können Sie folgende Faktoren erreichen:

- ✓ **Übersichtlicher Programmaufbau**: Der in sich geschlossene Codeblock einer Funktion kann mehrfach und unabhängig vom Fortschritt des Programms aufgerufen werden. Damit bietet es sich geradezu an, häufig verwendete Routinen in einer Funktion zusammenzufassen. Damit Sparen Sie viele Zeilen Programmcode, und das gesamte Programm wird transparenter und übersichtlicher.
- ✓ **Modularisierung und Wiederverwendbarkeit**: Ein weiterer Vorteil liegt in der einfachen Wiederverwendbarkeit und damit der Modularisierung von Code. Code, der in langer Liste eingebunden ist, kann meist nur schwer herausgefiltert und für andere Programme wiederverwendet werden. Eine Funktion hingegen lässt sich leicht an einer Stelle deklarieren oder definieren und diese aus verschiedenen Skripts in einem Film aufrufen. Mit der Verwendung von einheitlichen Namen für Variable schaffen Sie leicht überschaubare Schnittstellen zu diesen Funktionen. Damit können Sie zukünftige Programme in Modulbauweise zusammenstellen und verkürzen die Entwicklungszeit erheblich.

Definition einer Funktion

Eine `Function`-Prozedur ist eine Folge von Visual Basic-Anweisungen, die durch die Anweisungen `Function` und `End Function` eingeschlossen sind. Eine `Function`-Prozedur ähnelt einer `Sub`-Prozedur, kann aber auch einen Wert zurückgeben. Eine `Function`-Prozedur kann Argumente, wie z.B. `Konstanten`, `Variablen` oder Ausdrücke verwenden, die über die aufgerufene Prozedur übergeben werden. Wenn eine `Function`-Prozedur über keine Argumente verfügt, muß deren `Function`-Anweisung ein leeres Klammernpaar enthalten. Eine Funktion gibt einen Wert zurück, indem ihrem Namen ein Wert in einer oder mehreren Anweisungen der Prozedur zugewiesen wird.

Syntax:

```
[Public | Private] [Static] Function Name [(ArgListe)] [As Typ]
      [Anweisungen]
```

```
      [Name = Ausdruck]
      [Exit Function]
      [Anweisungen]
      [Name = Ausdruck]

End Function
```

Im folgenden Beispiel berechnet die `Celsius`-Funktion Celsius-Grade aus Fahrenheit-Graden. Wenn die Funktion aus der `Tempberechnen`-Prozedur aufgerufen wird, wird eine den Argumentwert enthaltende Variable der Funktion übergeben. Das Ergebnis der Berechnung wird an die aufrufende Prozedur zurückgegeben und in einem Meldungsfeld angezeigt.

```
Sub Tempberechnen()
   temp = InputBox(Prompt:= "Geben Sie die Temperatur in Fahrenheit ein.")
   MsgBox "Die Temperatur entspricht " & Celsius(temp) & " Grad Celsius."
End Sub

Function Celsius(fGrad)
      Celsius = (fGrad - 32) * 5 / 9
End Function
```

Sub-Anweisung

Prozedur ist eine benannte Folge von Anweisungen wie eine Funktion, die als Einheit ausgeführt werden. `Function`, `Property` und `Sub` sind zum Beispiel Prozedurtypen. Der Name einer Prozedur wird immer auf Modulebene definiert. Der gesamte ausführbare Code muss in einer Prozedur enthalten sein. Prozeduren können nicht in andere Prozeduren eingesetzt werden. `Sub`-Anweisung deklariert den Namen, die Argumente und den Code für den Rumpf einer Sub-Prozedur.

Syntax:

```
[Private | Public] [Static] Sub Name [(ArgListe)]
      [Anweisungen]
      [Exit Sub]
      [Anweisungen]

End Sub
```

`Sub`-Prozeduren sind standardmäßig öffentlich, wenn sie nicht explizit mit `Public` oder `Private` festgelegt werden. Wird `Static` nicht angegeben, so bleiben die Werte lokaler Variablen zwischen den Aufrufen nicht erhalten. Eine `Sub`-Prozedur können Sie nicht mit `GoSub`, `GoTo` oder `Return` aufrufen oder verlassen.

In diesem Beispiel wird die `Sub`-Anweisung verwendet, um Namen, Argumente und Prozedurrumpf einer `Sub`-Prozedur zu definieren.

Beispiel 27)

```
Sub Einfuegen(Tabellenname As String, KomboBox As Object, Spaltennummer As Integer)
    Dim varEndwert As Integer ' Lokale Variable deklarieren.
    Dim i As Integer
    varEndwert = Worksheets(Tabellenname).Cells(1, 1).CurrentRegion.Rows.Count
    ' Anzahl der belegten Zellen berechnen.
    i = 0
    For i = 1 To varEndwert
```

```
        KomboBox.AddItem Worksheets(Tabellenname).Cells(i, Spaltennummer).Value
    Next i
End Sub
```

In diesem Beispiel wird die `Sub`-Anweisung verwendet, um Prozedurrumpf für die Zuordnung an einer Schaltfläche oder in einem Programmablauf zu definieren.

Beispiel 28)

```
'Löscht den Inhalt der Eingabefelder und Kombinationsfeld im Dialogblatt frmZins
Sub Maskeloeschen()
    Me.cboPerson.Clear
    Me.cboPerson.Text = ""
    Me.txtAnlage.Text = ""
    Me.txtKapital.Text = ""
    Me.txtProzent.Text = ""
    Me.cboLaufzeit.Text = ""
End Sub
```

Weitere Informationen über die Prozeduren finden Sie in der Visual Basic-Hilfe unter dem Thema "Anweisungen in Visual Basic".

4 Dialogblatt (Userform)

Für die Programmsteuerung ist es oft nötig, Meldungen an Benutzer auszugeben bzw. Eingaben des Benutzers ins Programm zu übernehmen. Diese Aufgaben können vom System zur Verfügung gestellte Dialogfunktionen oder individuell durch den Benutzer definierte Oberflächen übernehmen.
Die Handhabung der System - Elemente wird im folgende näher beschrieben.

4.1 MsgBox

Die in den bisherigen Beispielen 5 und 10 im Abschnitt 3. 5 schon öfter benutzte MSGBOX ist eine den Programmablauf unterbrechende Funktion, die eine Meldung in einem Dialogfeld anzeigt und auf die Auswahl einer Schaltfläche wartet.
Die Funktion hat die Syntax:

```
MsgBox (Prompt [, Buttons][,Title][,Helpfile][,Context]
```

PROMPT	ist ein Zeichenfolgeausdruck mit der maximalen Länge von 1024 Zeichen, der als Meldung im Dialogfeld der MSGBOX erscheint. Soll der Meldungstext aus mehreren Zeilen bestehen, müssen die Zeilen durch manuelles Einfügen von Zeilenumbrüchen mittels der Funktion CHR (Zeichen CHR(13)) umbrochen werden.
BUTTONS	ist ein numerischer Ausdruck mir einem kombinierten Wert, der die Anzahl und den Typ der Schaltflächen, ein evtl. verwendetes Symbol, die aktivierte Schaltfläche und die Bindung des Dialogfeldes.
TITEL	ist ein Zeichenfolgeausdruck, der als Titel der Dialogbox erscheinen soll.
HELPFILE	Nur in Verbindung mit CONTEXT zu verwenden – definiert die kontextbezogene Hilfedatei für das Dialogfeld-
CONTEXT	Numerischer Ausdruck, der dem Hilfethema in der unter HELPFILE zugeordneten Hilfedatei zugeordnet ist

Die einfachste Form der Anwendung ist die Angabe der Funktion nur mit PROMPT:

Beispiel 29)

```
MsgBox "Sie müssen einen Eintrag wählen!"
MsgBox Preis * 1,19
MsgBox Worksheets("tblZinsEIN").Cells(2, 2).Value + 5
```

Wird das Argument BUTTONS verwendet, müssen entweder numerische Werte oder Konstanten zur Definition der Schaltflächen angegeben werden:

Schaltflächen:

Wert	Konstante	Funktion
0	vbOKOnly	(Voreinstellung) Schaltfläche OK erzeugen
1	vbOKCancel	OK und ABBRECHEN erzeugen
2	vbAbortRetryIgnore	ABBRECHEN, WIEDERHOLEN IGNORIEREN erzeugen
3	vbYesNoCancel	JA, NEIN, ABBRECHEN erzeugen
4	VbYesNo	JA, NEIN erzeugen
5	VbRetryCancel	WIEDERHOLEN, ABBRECHEN erzeugen

Dialogfeldsymbole:

Wert	Konstante	Funktion
16	vbCritical	Stop – Symbol
32	vbQuestion	Fragezeichen - Symbol
48	vbExclamation	Ausrufezeichen – Symbol
64	vbInformation	Info - Symbol

Aktivierte Schaltflächen:

Wert	Konstante	aktivierte Schaltfläche
0	vbDefaultButton1	erste Schaltfläche
256	vbDefaultButton2	zweite Schaltfläche
512	vbDefaultButton3	dritte Schaltfläche

Bindung (Modalverhalten) des Dialogfeldes:

Wert	Konstante	Funktion
0	vbApplicationModal	Gebunden an die Anwendung. Die aktuelle Anwendung kann nur fortgesetzt werden, wenn der MSGBOX – Dialog beendet wird. Alle anderen Anwendungen sind nicht betroffen.
256	vbSystemModal	Systemgebunden – alle Anwendungen werden angehalten, bis der MSGBOX – Dialog beendet ist.

Bei der Definition der Schaltflächen und Symbole können entweder die Wert oder die Konstanten benutzt werden.
Soll beispielsweise ein Dialogfeld mit einer JA-, einer NEIN – Schaltfläche, versehen mit dem Fragezeichen – Symbol und der aktivierten NEIN – Schaltfläche erzeugt werden, geschieht es in der folgenden Form:

```
vbYesNo + vbQuestion + vbDefaultButton2
```

Oder

```
4 + 32 + 256
```

oder einfach

```
292
```

wobei die letzte Version die Summe der Einzelwerte ist. Hier ist allerdings kaum erkennbar, was genau definiert wurde. Ein Dialogfeld mit einer JA-, einer NEIN – Schaltfläche, versehen mit dem Fragezeichen – Symbol und der aktivierten NEIN – Schaltfläche, welches die Meldung *„Soll der Datensatz gelöscht werden?“* anzeigt und den Titel *„Datensatz löschen“* besitzt, kann durch die folgende Anweisung erzeugt werden:

```
MsgBox "Soll der Datensatz gelöscht werden?", vbYesNo + vbQuestion + _
vbDefaultButton2, "Datensatz löschen"
```

Soll der durch das Dialogfeld gelieferte Rückgabewert ausgewertet werden, muß die Funktionsschreibweise (mit Argumentenklammern!) benutzt werden:

```
Ergebnis = MsgBox("Soll der Detnsatz gelöscht werden", vbYesNo +  _
vbQuestion + vbDefaultButton2, "Datensatz löschen")
```

Der zugewiesene Rückgabewert der Variablen Ergebnis kann anschließend ausgewertet werden.
Die Auswertung des Rückgabewertes wird meistens in einer Abfrage realisiert:

```
If Ergebnis = vbYes then....
```

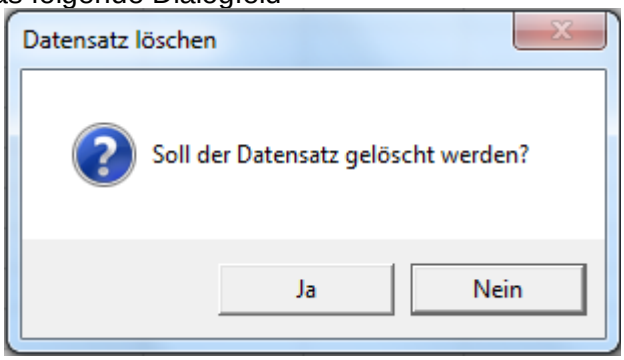
Oder

```
If Ergebnis = 6 then....
```

Der Umweg über die Variable kann gespart werden, wenn in die Abfrage die MSGBOX- Funktion aufgenommen wird:

```
Private Sub cmdloeschen_Click()
   Meldung = "Soll der Datensatz gelöscht werden?"
   Titel = "Datensatz löschen"
   Stil = vbYesNo + vbQuestion + vbDefaultButton2
   Ergebnis = MsgBox(Meldung, Stil, Titel)
   If Ergebnis = vbYes Then
       Worksheets("tblZinsEin").Rows(varWelcherEintrag).Delete Shift:=x1up
   Else
       Exit Sub
   End If
End Sub
```

Das Beispiel erzeugt das folgende Dialogfeld



4.2 Die Methode InputBox

Die Methode INPUTBOX unterscheidet sich von der Funktion INPUTBOX vor allem dadurch, dass sie eine Möglichkeit bittet, den Typ des eingegebenen Wertes zu definieren. Hierzu wird die Syntax um das Argument TYPE erweitert:

```
Object.InputBox(Prompt [,Titel][,default][,Left][,Top]
[,helpfile][,context][,Type])
```

Wird dieses Argument nicht angegeben, so wird der Eingabewert als Zeichenfolge interpretiert und bei der Übergabe an eine Zelle je nach Form als Zahl, Text, boolscher Wert oder Formel interpretiert. Dem Argument TYPE können folgende Werte zugewiesen werden:

Wert	Typ
0	Formel
1	Numerisch
2	Text
4	Logisch
8	Zellbezug (Bereichs – Objekt)
16	Fehlerwert (z.B. #NV)
64	Wertematrix

Der definierte Wert des Arguments kann sich auch aus einer Summe von Werten ergeben (siehe z.B. MSGBOX).

Die Argumente LEFT und TOP definieren im Unterschied zu XPOS und YPOS der Funktion jeweils den Abstand des Dialogfeldes in Points zum linken bzw. rechten Rand des Bildschirms

```
ActiveCell.Value = Application.InputBox("Bitte Anzahl Studenten angeben!", _
```

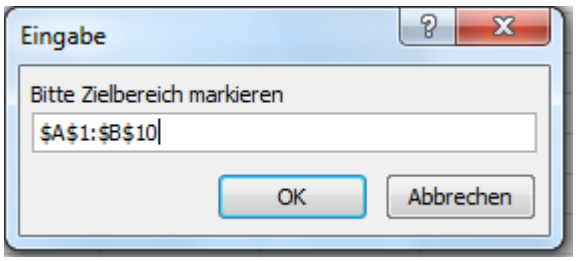
```
"Studentenstatistik", 0, , , , 1)
```

Ohne Angabe von OBJEKT (hier *Application*) geht VBA davon aus, dass die Funktion INPUTBOX benutzt wird und gibt eine Fehlermeldung wegen falscher Anzahl Argumente aus!

Wird für das Argument TYP der Wert 8 gesetzt, erlaubt die als Methode aktivierte INPUTBOX die Auswahl einer Zelle oder eines Zellbereichs per Maus anstelle einer manuellen Eingabe in das Eingabefeld. Die Methode gibt eine Range – Objekt zurück, welches mit Werten belegt werden kann.
Das folgende Beispiel demonstriert die Anwendung der InputBox – Methode zum Füllen markierter Zellbereiche mit Werten. Über das Dialogfeld wird der Benutzer aufgefordert, einen Zellbereich zu markieren. Dieser wird über die an das von INPUTBOX zurückgegebene RANGE – Objekt angehängte VALUE – Eigenschaft, mit einem Wert belegt.
Mit einem ähnlichen Verfahren können Zellbereiche mit identischen oder (soweit beispielsweise in einer Schleife erzeugt) unterschiedlichen Werten gefüllt werden. Insbesondere sind als einzutragende Werte auch Formeln möglich.

```
Private Sub UserForm_Initialize()
    Anzahl = 666
    Application.InputBox("Bitte Zielbereich markieren",,,,,,,8).Value = Anzahl
End Sub
```

Die Anweisungen des Beispiels erzeugen das Dialogfeld (hier schon nach der Markierung des Zellbereichs:



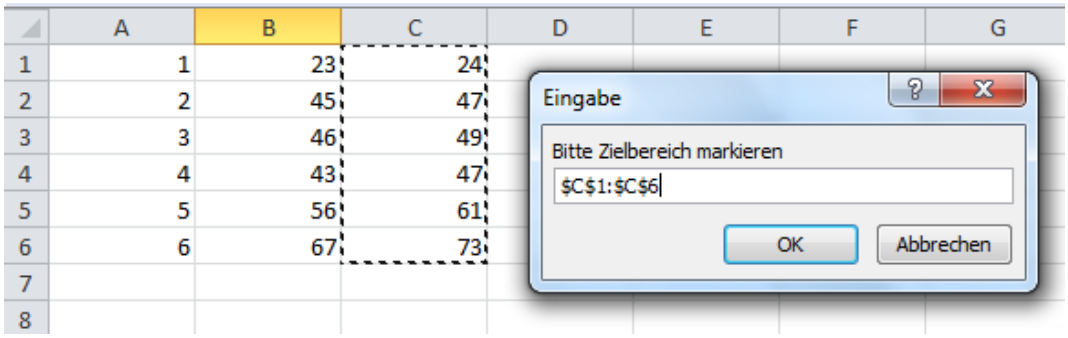
Nach Bestätigung mit Ok wird den Zellen des markierten Bereichs der gewünschte Wert zugewiesen:

	A	B
1	666	666
2	666	666
3	666	666
4	666	666
5	666	666
6	666	666
7	666	666
8	666	666
9	666	666
10	666	666
11		

Wird nach dem gleichen Verfahren versucht, eine Formel in einen markierten Bereich einzutragen, und enthält diese Formel relative Zelladressen, so werden diese angepaßt. Im folgenden Beispiel wurde die Formel in den markierten Bereich C1:C7 automatisch eingetragen. Ergebnis – siehe unten:

```
Private Sub UserForm_Initialize()
```

```
    Anzahl = "=A1+B1"
    Application.InputBox("Bitte Zielbereich markieren",,,,,,, 8).Value = Anzahl
End Sub
```



Soll die Relativierung der Adressen vermieden werden, müssen an entsprechenden Stellen der Formeln absolute Zellbezüge (Adressen) verwendet werden.

4.3 Steuerelemente aus *Selbstdefinierte Dialoge - UserForm*

Für den Aufbau eines Userdialogs wird in der VBA – Entwicklungsumgebung über die Funktionskombination EINFÜGEN / USERFORM ein leeres Dialogformular (Abb. 4.3) erzeugt.

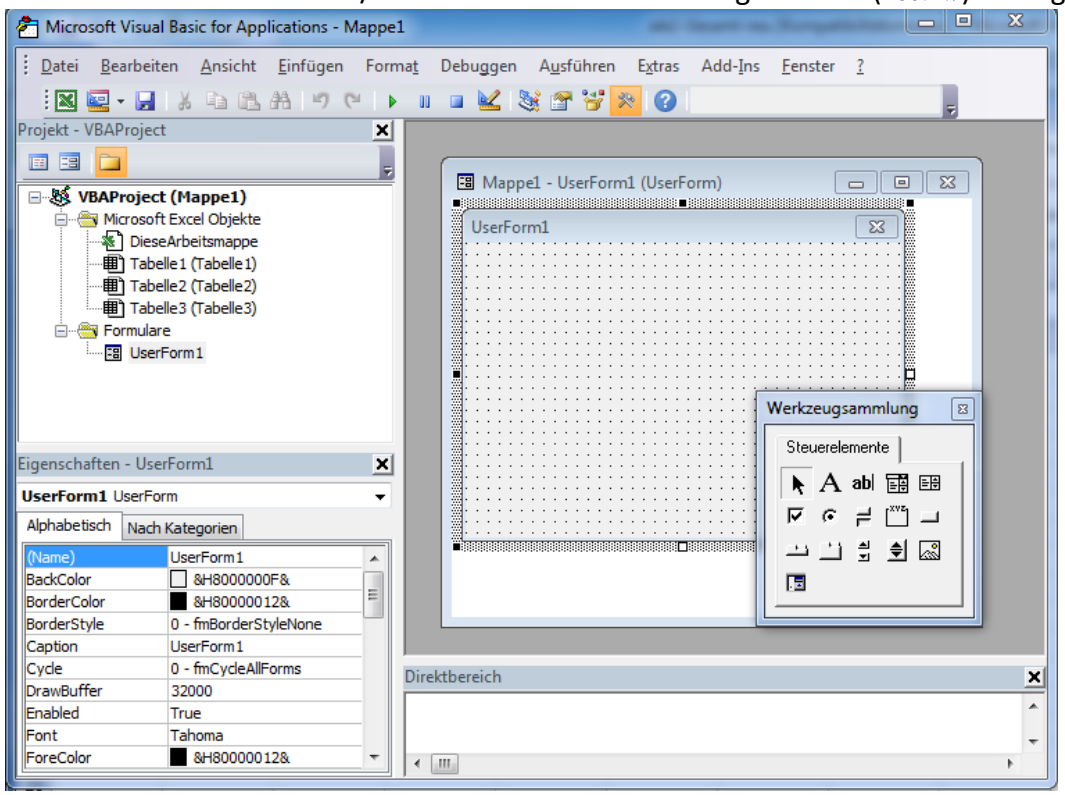


Abb. 4.3 leere Userform

Gleichzeitig wird die für den Formularaufbau benötigte Werkzeugsammlung eingeblendet.

4.3.1 Einfügen der Elemente ins Userformblatt

Die Steuerelemente werden in der Symbolleiste angeklickt und bei gedrückter linker Maustaste ins Userformblatt eingetragen (z.B. Rahmen zeichnen, nach dem Loslassen der Maustaste wird das Steuerelement in der eingezeichneten Größe angezeigt).

Die Bedeutung der Symbole (von Links nach rechts und von Oben nach Unten):

Objekte auswählen	**Zur Auswahl der Objekte bzw. Steuerelemente in einem Formular.**
Beschriftungsfeld (LABEL) Oder Bezeichnungsfeld	Zu Hinweise, Beschriftungen oder kurze Anleitungen auf einem Formular.
Textfeld (TEXTBOX)	Zur Aufnahme von Eingaben während der Laufzeit eines Programms.
Kombinationsfeld (COMBOBOX)	Kombination von Listenfeld und Textfeld.
Listenfeld (LISTBOX)	Definition von Auswahllisten.
Kontrollfeld (CHECKBOX)	Kann die Zustände aktiv / inaktiv annehmen. Für die Aufnahme von Optionen zu verwenden. Mehrere Kontrollfelder in einem Dialogblatt sind voneinander unabhängig.
Optionsfeld (OPTIONBUTTON)	Ähnlich der CHECKBOX . Kann allerdings zu Gruppen aus mehreren Elementen zusammengefaßt werden, in denen nur ein Element aktiv sein muß. Wird ein Element aktiviert, deaktiviert die Aktion ein anderes.
Umschaltfläche (TOGGLEBUTTON)	Wie Schaltfläche, jedoch zur Benutzung als Auswahl von mehreren alternativen Schaltflächen (aktiv = abgesenkt).
Rahmen (FRAME)	Zur Erstellung einer Optionsgruppe oder auch zur Gruppierung der Steuerelemente.
Schaltfläche (COMMANDBUTTON)	Schalterschaltfläche für dein Ereignis
Register (TABSTRIP)	Um inhaltlich zusammenhängende Steuerelemente in Gruppen mit jeweils unterschiedlichen Informationen zusammenzustellen und anzuzeigen.
Multiseiten (MULTIPAGE)	Zur Zuordnung der unterschiedlichen Kategorien, wenn mit größeren Mengen an Informationen gearbeitet wird.
Bildlaufleiste(SCROLLBAR)	Für die Eingabe numerischer Werte über Inkrement- und Dekrementschalter, Verschieben einer Schaltfläche oder Anklicken der Innenfläche.
Drehfeld (SPINBUTTON)	Für die Eingabe numerischer Werte über Inkrement- und Dekrementschalter.
Bildfeld (PICTURE) oder Anzeige	Feld zur Aufnahme einer Grafik.

4.3.2 Eigenschaften der Elemente

Die Steuerelemente können mit wesentlich mehr Eigenschaften ausgestattet werden als die der Formularleiste. Die Eigenschaften können per Dialogfenster (Abb. 4.3.2) oder direkt im Programm geändert werden.
Der Eigenschaften – Dialog wird aufgerufen, indem aus dem nach dem Klick mit der rechten Maustaste auf ein Steuerelement erscheinenden Auswahlmenu die Position EIGENSCHAFTEN gewählt wird oder im Menu EINSICHT den Menupunkt EIGENSCHAFTENFENSTER oder auch Klicken auf die Taste F4 gewählt wird.

Die Listen der daraufhin erscheinenden Eigenschaften von Steuerelementen sind unterschiedlich lang. Einige der Eigenschaften (Größe, Position) sind einfacher per Mausklick einzustellen, andere sollten über das Dialogfeld definiert werden.
Die Steuerelemente weisen eine Reihe gemeinsamer Eigenschaften auf, die hier vorab vorgestellt werden sollen. Sie können (neben vielen anderen) auch im Objektkatalog beim jeweiligen Control – Objekt eingesehen werden. Diese Eigenschaften können sowohl über den Eigenschafts – Dialog, als auch insbesondere zur Laufzeit eines Programms zugewiesen bzw. verändert werden.

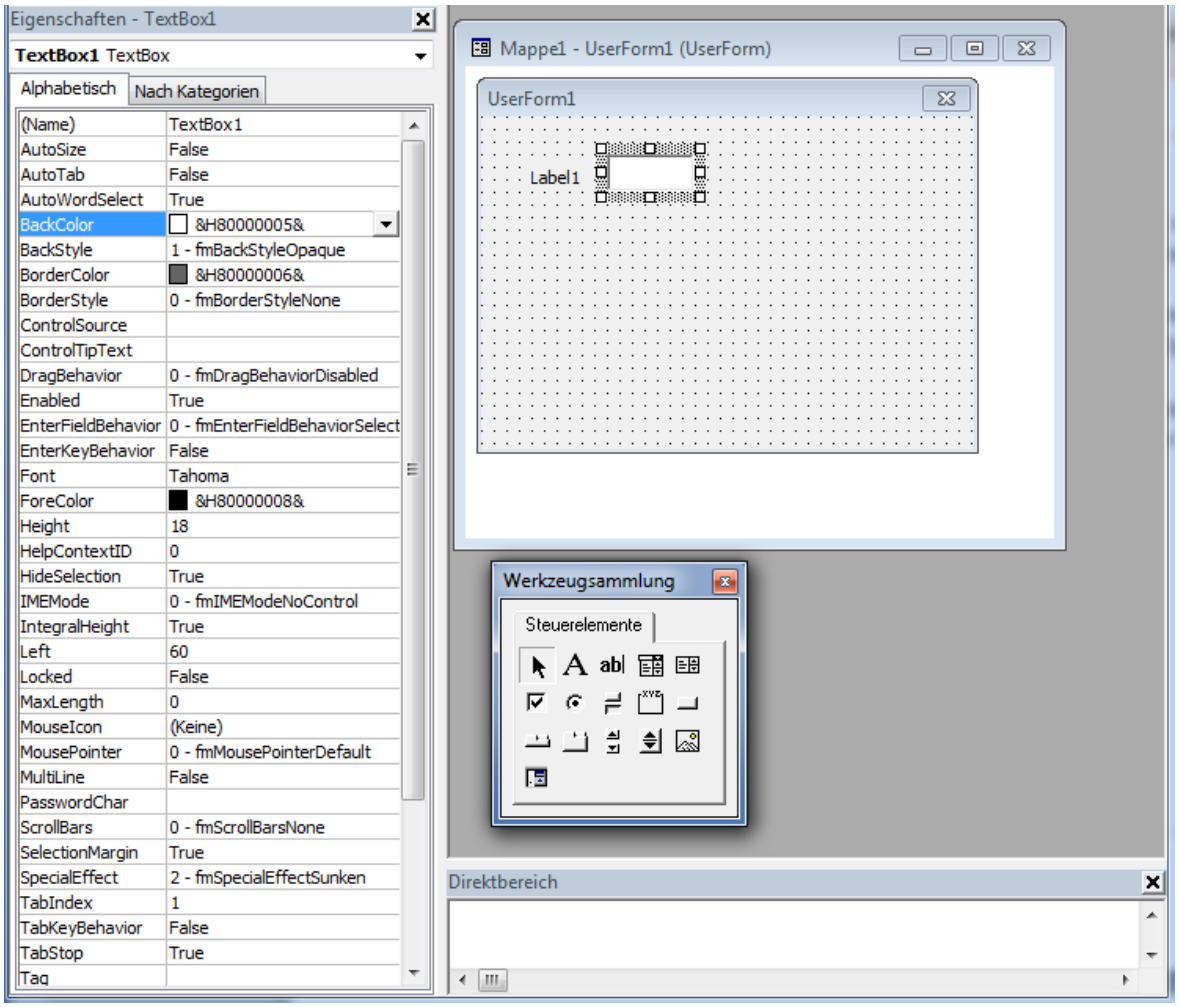


Abb. 4.3.2 Eigenschaften am Beispiel eines Textfeldes

Gemeinsame Eigenschaften	
NAME	Name des Steuerelements. Muss eindeutig sein Ohne Namen läßt sich das Steuerelement nicht im Programm verwenden. Excel vergibt automatisch einen Namen, sobald ein Element im Dialogformular eingesetzt wird. Der Name kann verändert werden, um aussagefähige individuelle Namen benutzen zu können.
CAPTION	Eigenschaft für Schalter, Options- und Kontrollfelder. Steht für die Beschriftung der Steuerelemente.
TEXT	Eigenschaft, die den Inhalt von Steuerelementen steuert, z.B. Inhalt von Textfeldern.
VALUE	Gibt den Zustand oder Inhalt eines angegebenen Steuerelements an, z.B. den Inhalt eines Textfeldes oder den Zustand einer Schaltfläche oder Optionsfeldes usw.
CANCEL	= TRUE, wenn das Steuerelement durch Esc verlassen werden kann. Wird normalerweise verwendet, damit die nicht abgeschlossene Änderungen abgebrochen werden können und den vorhergehenden Zustand eines Formulars wieder hergestellt werden kann.
CONTROLTIPTEXT	Gibt den Text an, der angezeigt wird, wenn der Benutzer den Mauszeiger eine Weile über ein Steuerelement hält, ohne zu klicken.
CONTROLSOURCE	Stellt die Verbindung zu einer Zelle im Tabellenblatt her. Bezeichnet die Datenposition, mit der die VALUE - Eigenschaft eines Steuerelements festgesetzt oder gespeichert werden kann.
DEFAULT	= TRUE, wenn das Element durch ENTER ausgewählt werden kann: Legt die Standardbefehlsschaltfläche eines Formulars fest.
LINKEDCELL	Bildet die Verbindung zwischen der Eigenschaft VALUE bzw. TEXT eines Elements und einer

	Zelle im Tabellenblatt. Ein Eintrag in Element wird an die über diese Eigenschaft zugeordnete Zelle weitergegeben und umgekehrt.
ROWSOURCE	Stellt die Verbindung zu einem Zellbereich her. Gibt die Quelle an, die eine Liste für ein Kombinationsfeld-Steuerelement (COMBOBOX) oder Listenfeld-Steuerelement (LISTBOX) zur Verfügung stellt.
TABINDEX	Gibt die Position eines einzelnen Objekts in der Aktivierreihenfolge des Formulars an.
TABSTOP	= TRUE, wenn das Element mit der Tab – Taste ausgewählt werden kann. Zeigt an, ob ein Objekt den Fokus erhalten kann, wenn die TAB - Taste drückt wird.
VISIBLE	= TRUE, wenn das Element sichtbar werden soll.
Gemeinsame Eigenschaften für die Stilbestimmung	
BACKCOLOR	Bestimmt die Hintergrundfarbe eines Steuerelements (für Text-, Listen-, Options- und Kontrollfelder).
BORDERCOLOR	Bestimmt die Rahmenfarbe (für Text-, Listen-, Options- und Kontrollfelder).
BORDERSTYLE	Betsimmt den Rahmentyp (für Text-, Listen-, Options- und Kontrollfelder).
FONT	Schriftart und Schriftattribute.
FORECOLOR	Vordergrundfarbe des Elements.
SHADOW	Darstellung eines Schattens.
HEIGHT	Höhe des Elements.
WIDTH	Breite des Elements
SPECIALEFFEKT	Zuweisung von 2D / 3D – Effekten.
Gemeinsame Methoden	
SETFOCUS	Setzt den Eingabefokus auf das Element.

4.3.3 Das Bezeichnungsfeld (Label)

Das Bezeichnungsfeld wird zur Beschriftung eines Dialogs benutzt. Es wird neben Steuerelementen plaziert, um sie zu Beschriften bzw Hinweise zu deren Benutzung geben
zu können.

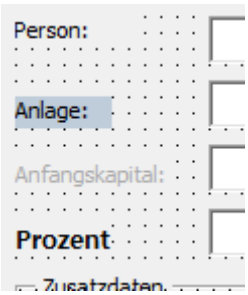


Abb. 4.3.3 Unterschiedliche Formen des LABEL-Feldes

In diesem Skript wurden die Namen der Bezeichnungsfelder nicht geändert sondern vom System übernommen.

Wichtigste Eigenschaften:

- ✓ Der darzustellende Text wird über die Eigenschaft CAPTION definiert.
- ✓ Bei mehrzeiligen Texten muß die Eigenschaft WORDWRAP auf TRUE gesetzt werden.
- ✓ Die Textausrichtung wird über die Eigenschaft TEXTALIGN geregelt.
- ✓ Soll sich die Größe des Feldes an die Länge des Textes anpassen, muß AUTOSIZE auf
- ✓ TRUE gesetzt werden.
- ✓ Schriftart und –farbe werden über FONT, BACKCOLOR bzw. FORECOLOR eingestellt.
- ✓ Die Art der Umrandung kann über BORDERSTYLE und BORDERCOLOR definiert werden.
- ✓ Mit PICTURE und PICTUREPOSITION kann eine Bitmap eingesetzt werden.

Die Einstellung der Eigenschaften eines Bezeichnungsfeldes ist zwar auch zur Laufzeit des Programms möglich, es empfiehlt sich aber es nur dann im Programm zu tun, wenn Inhalte des Feldes dynamisch zur Laufzeit verändert werden sollen. Ansonsten empfiehlt es sich, die Eigenschaften im Dialogfenster festzulegen.

4.3.4 Schaltflächen, Wechselschaltflächen (CommandButton, ToggleButton)

Die Schaltflächen – Definition ist recht einfach. Sowohl die normalen Schaltflächen als auch die Umschaltbuttons (Wechselschaltflächen) besitzen gleiche Definitions – Eigenschaften. Der Unterschied besteht darin, daß die ToggleButtons nach dem Aktivieren (anklicken oder Enter – Taste betätigen) solange im gedrückten Zustand verbleiben, bis sie wieder angeklickt werden.

Abb. 4.3.4 Unterschiedliche Formen des Schaltflächen

Wichtigste Eigenschaften:

- ✓ Der darzustellende Text wird über die Eigenschaft CAPTION definiert.
- ✓ Mit PICTURE kann eine Bitmap – Grafik statt Beschriftung eingefügt werden.
- ✓ Im Falle einer eingefügten Grafik kann mit CONTROLTIPTEXT ein gelber Infotext definiert werden.
- ✓ Mit PICTUREPOSITION wird die Position der Grafik festgelegt.
- ✓ Soll die Schaltflächengröße an den Inhalt angepaßt werden, geschieht es über AUTOSIZE.
- ✓ Der aktuelle Zustand kann über VALUE abgefragt werden.

4.3.5 Textfelder (TextBox)

Textfelder ermöglichen die Eingabe von Texten. Die Eigenschaften dieser Felder sind größtenteils mit den der Bezeichnungsfelder (Label) identisch, weswegen an dieser Stelle auf die nochmalige Beschreibung identischer Eigenschaften verzichtet wird.

Abb. 4.3.5 Unterschiedliche Formen des Textfeldes

Wichtigste zusätzliche Eigenschaften:

- ✓ Mehrzeilige Textfelder werden mit MULTILINE definiert.
- ✓ Bei mehrzeiligen Textfeldern können mit SCROLLBARS Laufleisten eingeblendet werden, wenn der Text über die definerte Breite hinausgeht.
- ✓ ENTERKEYBEHAVIOR = TRUE bewirkt einen Zeilenumbruch bei Betätigung der ENTER – Taste (MULTILINE muß auf TRUE gesetzt sein).
- ✓ Mit ENTERFIELDBEHAVIOR = 0 wird beim Aktivieren des Textfeldes der gesamte Inhalt markiert, was die Neueingaben, insbesondere bei einzeiligen Feldern praktischer macht.
- ✓ WORDWRAP bewirkt einen Zeilenumbruch, wenn der Text den rechten Rand erreicht (erfordert MULTILINE = TRUE).
- ✓ Die Textausrichtung wird mit TEXTALIGN definiert.
- ✓ Der Zugriff auf den Inhalt erfolgt über TEXT.
- ✓ Die Anzahl Zeilen kann mit LINECOUNT, die aktuelle Zeile mit CURLINE die Anzahl Zeichen durch LEN(FELDNAME.TEXT) ermittelt werden.

- ✓ Soll das Textfeld als Passwordfeld (Passwordeingabe) dienen, können mit PASSWORDCHAR Zeichen definiert werden, die statt des eingegebenen Textes erscheinen.
- ✓ Die Auswahl SELECTMARGIN = TRUE erzeugt einen Leerraum am linken Rand, was eine bequeme Markierung von zeilen in mehrzeiligen Textfeldern möglich macht.

Auf den markierten Text kann über SELTEXT zugegriffen werden. Die Eigenschaften SELSTART und SELLENGHT geben die Position des ersten markierten Zeichens und die Länge der Markierung zurück. Damit kann per Programm Markiert oder eine Textmarkierung manipuliert werden:

```
With Textfeld
      .SelLength = 0     'Löschen der Markierung
      .SelStart = 0      '
      .SelLength = 8     'die ersten 8 Zeichen markieren
      .SelText = ""      'markierter Text wird gelöscht
      .SelStart = 15     'Cursor auf neue Position
      .SelText = "ein"   'Zeichenkette an neuer Position einfügen
End With
```

Die Methoden CUT bzw. COPY übertragen den markierten Text in die Zwischenablage (Ausschneiden, Kopieren), die Methode PASTE überträgt den Inhalt der Zwischenablage in den markierten Bereich des Textes.

4.3.6 Listen, Kombinationsfelder (ListBox, ComboBox)

Listenfelder erlauben Auswahlen von Alternativen ohne direkte Eingabe, Kombinationsfelder (Abb. 4.3.6) bieten neben der gleichen Technik zusätzlich noch als Kombination von Listen- und Textfeldeigenschaften Eingabemöglichkeiten an.

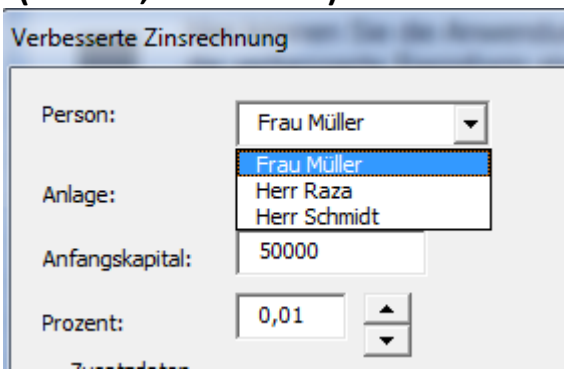


Abb. 4.3.6 Einfaches Kombinationsfeld

Wichtigste Eigenschaften:

- Der Zellbereich, aus dem die Werte für die Liste entnommen werden sollen wird über `Worksheets("tblZinsEIN ).Cells(1, 1).CurrentRegion.Rows.Count` definiert (in dem obigen Beispiel 5).
- Der Textinhalt des Kombinationsfeldes wird durch die Anweisung `If Me.cboPerson.Text = "" Then` abgefragt (in dem obigen Beispiel 10).
- Weitere Eigenschaften sind in der Bearbeitung

4.3.7 Drehfelder, Laufleisten (SpinButton,ScrollBar)

Bildlaufleisten und Drehfelder(Abb. 4.3.7) dienen dazu eine ganze Zahl aus einem vordefinierten Wertebereich zu selektieren. Der zulässige Zahlenbereich liegt im LONG – Zahlenraum (etwa + / - 2.109).

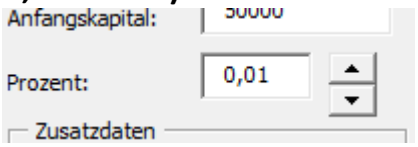


Abb. 4.3.7 Drehfeld

Bei Laufleisten kann der Wert durch das Anklicken einer der Inkrement- oder Dekrement Schaltflächen, das Verschieben eines Schiebers oder einen Klick in die Lauffläche verändert werden. Das Drehfeld ist eine „abgemagerte“ Variante der Laufleiste – es enthält nur eine Inkrement und eine Dekrement – Schaltfläche.

Wichtigste Eigenschaften:

- DELAY bestimmt die Verzögerung zwischen Klick und Ergebnis in Milisekunden.
- Die Grenzen des zulässigen Wertebereichs werden mit MIN / MAX festgelegt werden.
- ORIENTATION bestimmt die Ausrichtung des Steuerelements (vertikal / horizontal).
- Die Abmessungen des Schiebers einer Laufleiste werden mit PROPORTIONALTHUMP festgelegt. Steht der Wert auf TRUE, so ist die Schiebergröße umgekehrt proportional zur Größe des Wertebereichs. Bei großen Wertebereichen auf FALSE setzen, sonst ist der Schieber kaum sichtbar und bedienbar!
- SMALLCHANGE legt die Schrittweite der Wertänderung beim Klick auf die Inkrement bzw. Dekrement – Schaltflächen fest.
- Erzeugte Werte können über VALUE abgefragt werden.

4.3.8 Kontrollkästchen, Optionsfelder (CheckBox, OptionButton)

Kontrollkästchen (Abb. 4.3.8) eignen sich in für Ja / Nein – Entscheidungen. Der aktuelle Zustand wird durch ein Häkchen ✔ im quadratischen Fenster des Kontrollkästchens angezeigt.

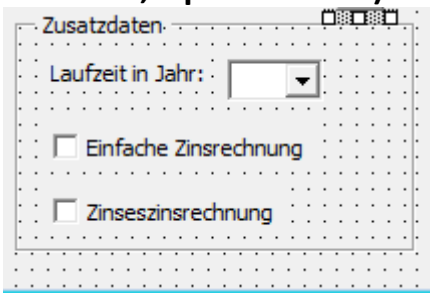


Abb. 4.3.8 Kontrollfelder

Wichtigste Eigenschaften:

- Der aktuelle Zustand kann über die Eigenschaft VALUE erfragt werden. Zulässige Werte sind FALSE, TRUE und NULL.
- Die Beschriftung kann über CAPTION festgelegt werden.
- Über die Eigenschaft GROUPNAME können Optionsfelder und Kontrollfelder gruppiert werden. Sollen mehrere Optionsfelder zu einer Gruppe gehören, wird bei allen unter GROUPNAME der gleiche Gruppenname angegeben.

A1) Aufgabe: Eine Datenbank Modelldatenblätter

Beschreibung

Bei dieser Aufgabe handelt es sich darum, die Daten von einer User form in eine Excel-Tabelle zu übertragen. Dabei sollen die Stammdaten wie *Produktgruppe, Baugruppe, Bausteinnummer, Bausteinbezeichnung, Beschreibung1* bis *Beschreibung6* und das dazugehöriges Bild mithilfe einer Userform in eine Ausgabetabelle eins zu eins übertragen werden.

Wichtige Funktionen

Aufbau einer Exceldatei zu einem automatisierten Modelldatenblatt mit folgenden Funktionen:

- Umgang mit den Tabellenblättern
- Umgang mit den Steuerelementen zur Steuerung der Daten
- Umgang mit einer Userform, um die Daten z.B. von einer Stammdatentabelle in eine Ausgabetabelle zu übertragen
- Zugriff auf die Zellen in einer Tabelle
- Umgang mit VBA-Syntax (u.a Variablen und deren Datentypen, Gültigkeitsbereich einer Variable, Fallunterscheidungen, Schleifen, Konstanten, Operatoren, eindimensionalen Arrays, benutzerdefinierten Prozeduren)
- Konzeption und Realisation der Algorithmen zu der Aufgabestellung

Vorgehensweise

Erstellen Sie eine Exceldatei mit dem Namen axxxxxxx.xLsm (xxxxxxx steht für Ihre Matrikelnummer). In dieser Aufgabe werden zwei Tabellen (tblBSTEIN als Stammdatentabelle und tblBSTAUS als Ausgabetabelle) und eine Userform mit dem Namen frmBasisform benötigt.
A1.1) Das Tabellenblatt (Tab A1.1a) kann tblBSTEIN heißen und beinhaltet Datensätze für die *Bausteinnummer*(Spalte A), *Bausteinbezeichnung*(Spalte B), *Produktgruppe*(Spalte C), *Baugruppe*(Spalte D), *Beschreibung1*(Spalte E) bis *Beschreibung6*(Spalte J).

	A	B	C	D	E	F	G	H	I	J
1	301	speltaschen mit	Hose	Basisform	- Bundfaltenhose, 2 Falten	- Vorderhose Doppelpaspeltaschen	- Hinterhose Doppelpaspeltaschen	- Saumaufschlag 35 mm	- mäßige Weiterzugabe	- Passformklasse (KI)
2	311	peltaschen ohne	Hose	Basisform	- Bundfaltenhose, 2 Falten	- Vorderhose Doppelpaspeltaschen	- Hinterhose Doppelpaspeltaschen	- Ohne Aufschlag	- mäßige Weiterzugabe	- Passformklasse (KI)
3	312	taschen mit Auf	Hose	Basisform	- Bundfaltenhose, 2 Falten	- Vorderhose Doppelpaspeltaschen	- Hinterhose Doppelpaspeltaschen	- Saumaufschlag 35 mm	- mäßige Weiterzugabe	- Passformklasse (KI)

Tab A1.1a tblBSTEIN

A1.2) Das Userformblatt (Abb. A1.2a) kann frmBasisform heißen und besteht aus den folgenden Elementen:

Pos.	Bezeichnung (Caption)	Steuerelement	Name
1	Bausteinnummer	Beschriftungsfeld	Nicht ändern
2	Keine	Kombinationsfeld	cboBstn

3	Bausteinbezeichnung	Beschriftungsfeld	Nicht ändern
4	Keine	Textfeld	txtBsbez
5	Produktgruppe	Beschriftungsfeld	Nicht ändern
6	Keine	Textfeld	txtPg
7	Baugruppe	Beschriftungsfeld	Nicht ändern
8	Keine	Textfeld	txtBg
9	Beschreibung 1	Beschriftungsfeld	Nicht ändern
10	Keine	Textfeld	txtBesch1
11	Beschreibung 2	Beschriftungsfeld	Nicht ändern
12	Keine	Textfeld	txtBesch2
13	Beschreibung 3	Beschriftungsfeld	Nicht ändern
14	Keine	Textfeld	txtBesch3
15	Beschreibung 4	Beschriftungsfeld	Nicht ändern
16	Keine	Textfeld	txtBesch4
17	Beschreibung 5	Beschriftungsfeld	Nicht ändern
18	Keine	Textfeld	txtBesch5
19	Beschreibung 6	Beschriftungsfeld	Nicht ändern
20	Keine	Textfeld	txtBesch6
21	Keine	Anzeige	imgBild
22	Ablegen	Befehlsschaltfläche	cmdAblegen
22	Löschen	Befehlsschaltfläche	cmdloeschen
23	Ausgeben	Befehlsschaltfläche	cmdAusgeben
24	Beenden	Befehlsschaltfläche	cmdBeenden

Erläuterung zu dem Programmablauf

A1.2.1) Beim Starten des Userformblatts (Abb. A1.2a) müssen alle Einträge der ersten Spalte von der Tabelle (Tab. A1.1a) in das Kombinationsfeld `cboBstn` eingefügt werden.

A1.2.2) Wenn ein Eintrag für die Bausteinnummer gewählt wird, muss der entsprechende Datensatz aus der Tabelle tblBSTEIN (Tab. A1.1a) in den Textfeldern und das dazugehörige Bild in dem Steuerelement imgBild erscheinen.

A1.2.3) Wenn der Benutzer einen neuen Eintrag in das Userformblatt einträgt und auf „Ablegen“ klickt, muss der neue Datensatz in die Tabelle (Tab. A1.1a) hinzugefügt werden und anschließend sortiert werden.

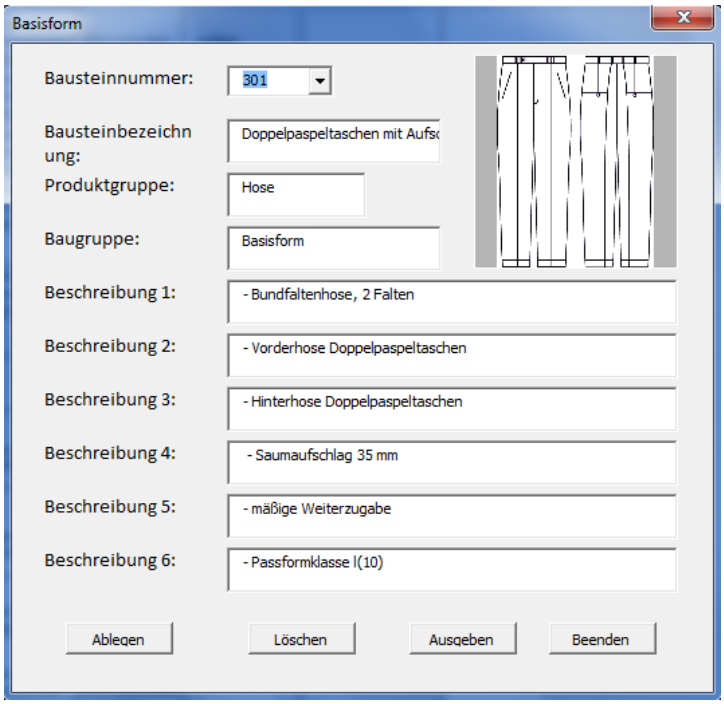


Abb. A1.2a frmBasisform

A1.2.4) Wenn der Benutzer einen Eintrag wählt, der schon in der Tabelle (Tab. A1.1a) existiert, muss der Datensatz beim Klicken auf „Ablegen“ ersetzt werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, dass der Datensatz ersetzt werden soll.)

A1.2.5) Wenn der Benutzer einen Eintrag wählt und auf die Befehlsschaltfläche „Löschen“ klickt, muss der Datensatz aus der Tabelle (Tab. A1.1a) gelöscht werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, dass der Datensatz gelöscht werden soll.)

A1.2.6) Wenn der Benutzer auf die Befehlsschaltfläche „Ausgeben“ klickt, müssen alle Daten von Userform (Abb. A1.2a) in die Ausgabetabelle(Tab. A1.3a) übertragen werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, wenn der Benutzer keinen Eintrag gewählt hat.)

A1.2.7) Die Befehlsschaltfläche „Beenden“ blendet die Userform aus.

A1.3) Erstellen Sie eine Tabelle(Tab. A1.3a) mit dem Namen tblBSTAUS. Die Werte für die *Produktgruppe*(Zeile 1 /Spalte B), die *Baugruppe*(Zeile 2 /Spalte B), die *Bausteinnummer* (Zeile 3 /Spalte B), die *Bausteinbezeichnung* (Zeile 5 /Spalte A), *Beschreibung1* (Zeile 5 /Spalte B) bis *Beschreibung6* (Zeile 10 /Spalte B) müssen von der Userform frmBasisform (Abb. A1.2a) in diese Tabelle bzw. in entsprechende Zellen übertragen werden. Das zu der Bausteinnummer entsprechende Bild muss in die Zelle (Zeile 5 /Spalte C) plaziert werden.

	A	B	C
1	**Produktgruppe:**	*Hose*	**Modelldaten**
2	**Baugruppe:**	*Basisform*	
3	**Bausteinnr.:**	*301*	
4	**Baustein-bezeichnung**	**Beschreibung**	**Zeichnung**
5	*Doppelpaspeltaschen mit Aufschlag*	*- Bundfaltenhose, 2 Falten*	
6		*- Vorderhose Doppelpaspeltaschen*	
7		*- Hinterhose Doppelpaspeltaschen*	
8		*- Saumaufschlag 35 mm*	
9		- mäßige Weiterzugabe	
10		*- Passformklasse I(10)*	
11			

Tab. A1.3a tblBSTAUS

Programmablaufplan

Der Programmablaufplan (Abb. 1.6a) bezieht sich auf die Abschnitte A1.2.1 – A1.2.7. Entwickeln und realisieren Sie bitte einen Programmablaufplan, in dem kurze Codes und vereinfachte Aktionen für den Benutzer ermöglicht werden.

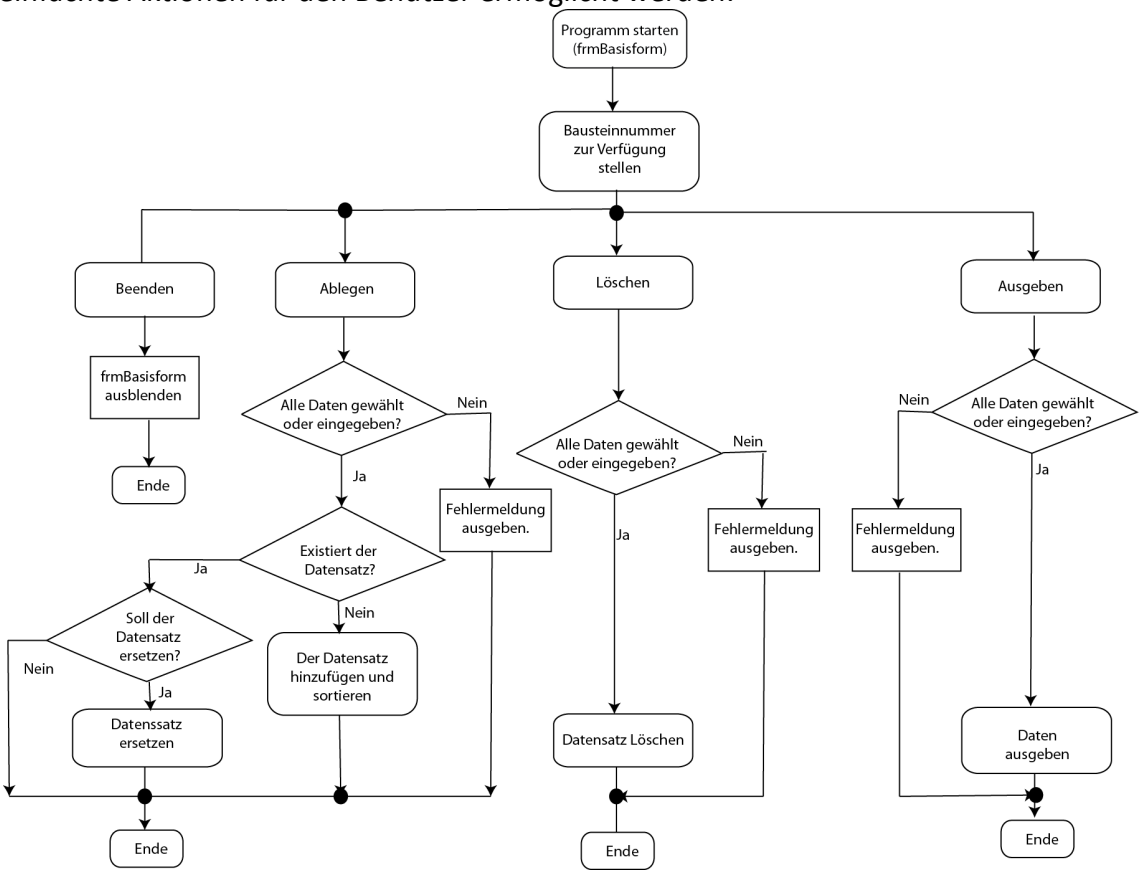


Abb. A1.6a Programmablaufplan zu Basisform

A1.6) Folgende VBA-Codes dienen dazu, in den Abschnitten A1.2.1-A1.2.7 beschriebenen Programmablaufplan zu implementieren:

```
Dim varWelcherEintrag As Integer
Dim varWelcherPath As String
Dim varMatrix(1 To 10) As Variant
Private Sub UserForm_Initialize()
    Dim varDatei As String
    Dim varDateiName As String
    varDatei = ActiveWorkbook.FullName
    varDateiName = Dir(varDatei)
    varWelcherPath = Left(varDatei, Len(varDatei) - Len(varDateiName)) + "Bilder\"
    Me.imgBild.Picture = LoadPicture(varWelcherPath & "blank.jpg")
    Maskeloeschen
    Einfuegen
End Sub
```

```
Sub Einfuegen()
    Dim varBausteinnr As Integer
    varBausteinnr = Worksheets("tblBSTEIN").Cells(1, 1).CurrentRegion.Rows.Count
    For i = 1 To varBausteinnr
        Me.cboBsnr.AddItem Worksheets("tblBSTEIN").Cells(i, 1).Text
    Next i
End Sub
```

```
Private Sub cboBsnr_Click()
    Dim varBild As Variant
    varWelcherEintrag = Me.cboBsnr.ListIndex + 1
    Me.txtBsbez.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 2).Value
    Me.txtPg.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 3).Value
    Me.txtBg.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 4).Value
    Me.txtBesch1.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 5).Value
    Me.txtBesch2.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 6).Value
    Me.txtBesch3.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 7).Value
    Me.txtBesch4.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 8).Value
    Me.txtBesch5.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 9).Value
    Me.txtBesch6.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 10).Value
    Dim fso
    Set fso = CreateObject("Scripting.FileSystemObject")
    If (fso.FileExists(varWelcherPath & Me.cboBsnr.Text & ".jpg")) Then
        Me.imgBild.Picture = LoadPicture(varWelcherPath & Me.cboBsnr.Text & ".jpg")
    Else
        varBild = Application.GetOpenFilename("JPG Files (*.jpg), *.gif")
        If varBild <> False Then
            Me.imgBild.Picture = LoadPicture(varBild)
            varWelcherPath = Left(varBild, Len(varBild) - Len(Me.cboBsnr.Text & ".jpg"))
        Else
            Me.imgBild.Picture = LoadPicture(varWelcherPath & "blank.jpg")
        End If
    End If
End Sub
```

```
Private Sub cboBsnr_Enter()
    Me.txtBsbez.Text = ""
    Me.txtPg.Text = ""
    Me.txtBg.Text = ""
    Me.txtBesch1.Text = ""
    Me.txtBesch2.Text = ""
    Me.txtBesch3.Text = ""
    Me.txtBesch4.Text = ""
    Me.txtBesch5.Text = ""
    Me.txtBesch6.Text = ""
    Me.imgBild.Picture = LoadPicture(varWelcherPath & "blank.jpg")
End Sub
```

```
Private Sub cboBsnr_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Me.txtBsbez.Text = ""
    Me.txtPg.Text = ""
    Me.txtBg.Text = ""
    Me.txtBesch1.Text = ""
    Me.txtBesch2.Text = ""
    Me.txtBesch3.Text = ""
    Me.txtBesch4.Text = ""
    Me.txtBesch5.Text = ""
    Me.txtBesch6.Text = ""
    Me.imgBild.Picture = LoadPicture(varWelcherPath & "blank.jpg")
End Sub
```

```
Private Sub cmdLoeschen_Click()
    Mldg = "Wollen Sie wirklich diesen Datensatz löschen?"
    Stil = vbYesNo + vbCritical + vbDefaultButton2
    Title = "Daten Löschen"
    If Me.cboBstnr.Text > "" Then
        Ergebnis = MsgBox(Mldg, Stil, Title)
        If Ergebnis = vbYes Then
            Worksheets("tblBSTEIN").Rows(varWelcherEintrag).Delete Shift:=xlUp
            Maskeloeschen
            Einfuegen
        End If
    Else
        MsgBox "Sie müssen zuerst einen Eintrag wählen!"
    End If
End Sub
```

```
Sub Maskeloeschen()
    Me.cboBsnr.Clear
    Me.txtBsbez.Text = ""
    Me.txtPg.Text = ""
    Me.txtBg.Text = ""
    Me.txtBesch1.Text = ""
    Me.txtBesch2.Text = ""
    Me.txtBesch3.Text = ""
    Me.txtBesch4.Text = ""
    Me.txtBesch5.Text = ""
    Me.txtBesch6.Text = ""
End Sub
```

```
Private Sub cmdAblegen_Click()
    Mldg = "Wollen Sie den alten Datensatz ersetzen?"
    Stil = vbYesNo + vbCritical + vbDefaultButton2
    Title = "Basisform"
    If Me.cboBsnr.Text = "" Then
        MsgBox "Sie müssen einen Eintrag wählen."
    Else
        If Me.cboBsnr.Text > "" And varWelcherEintrag = 0 Then
            Worksheets("tblBSTEIN").Activate
            Neueintragen
            Maskeloeschen
            Einfuegen
        End If
        If Me.cboBsnr.Text > "" And varWelcherEintrag > 0 Then
            If Me.cboBsnr.Text = Worksheets("tblBSTEIN").Cells(varWelcherEintrag, 1).Value Then
                Ergebnis = MsgBox(Mldg, Stil, Title)
                If Ergebnis = vbYes Then
                    Uebertragen
                    For i = 1 To 10
                       Worksheets("tblBSTEIN").Cells(varWelcherEintrag, i).Value = varMatrix(i)
                    Next i
                    Maskeloeschen
                    Einfuegen
                    Exit Sub
                Else
                    Exit Sub
                End If
            Else
                Neueintragen
                Maskeloeschen
                Einfuegen
            End If
        End If
    End If
End Sub
```

```
Sub Uebertragen()
    varMatrix(1) = Me.cboBsnr.Text
    varMatrix(2) = Me.txtBsbez.Text
    varMatrix(3) = Me.txtPg.Text
    varMatrix(4) = Me.txtBg.Text
    varMatrix(5) = Me.txtBesch1.Text
    varMatrix(6) = Me.txtBesch2.Text
    varMatrix(7) = Me.txtBesch3.Text
    varMatrix(8) = Me.txtBesch4.Text
    varMatrix(9) = Me.txtBesch5.Text
    varMatrix(10) = Me.txtBesch6.Text
End Sub
```

```
Sub Neueintragen()
```

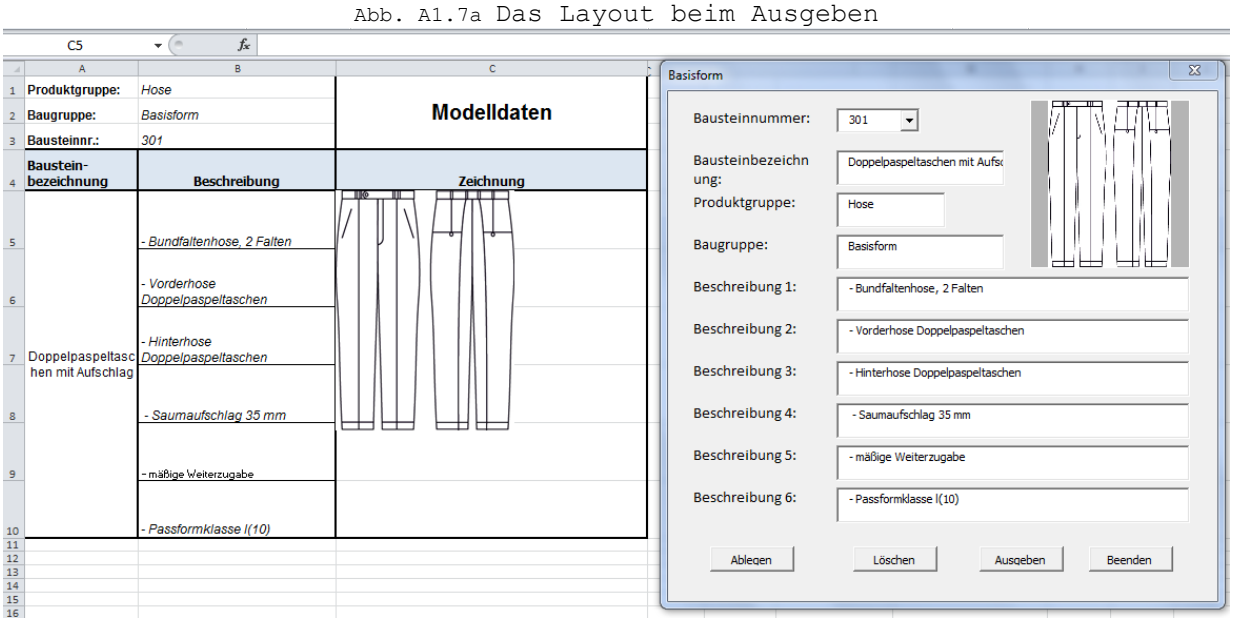
```
    Dim VarNeueZeile As Integer
    VarNeueZeile = Worksheets("tblBSTEIN").Cells(1, 1).CurrentRegion.Rows.Count + 1
    Uebertragen
    For i = 1 To 10
       Worksheets("tblBSTEIN").Cells(VarNeueZeile, i).Value = varMatrix(i)
    Next i
    With Worksheets("tblBSTEIN")
        .Range(.Cells(1, 1), .Cells(VarNeueZeile, 10)).Sort Key1:= _
            Worksheets("tblBSTEIN").Columns("A"), Header:=xlGuess
    End With
    varWelcherEintrag=VarNeueZeile
End Sub
```

```
Private Sub cmdAusgeben_Click()
    Dim varBild As Picture
    If Me.cboBsnr.Text = "" Then
        Beep
        MsgBox "Sie müssen einen Eintrag wählen."
    Else
        Worksheets("tblBSTAUS").Activate
        Worksheets("tblBSTAUS").Cells(3, 2).Value = Me.cboBsnr.Text
        Worksheets("tblBSTAUS").Cells(1, 2).Value = Me.txtPg.Text
        Worksheets("tblBSTAUS").Cells(2, 2).Value = Me.txtBg.Text
        Worksheets("tblBSTAUS").Cells(5, 1).Value = Me.txtBsbez.Text
        Worksheets("tblBSTAUS").Cells(5, 2).Value = Me.txtBesch1.Text
        Worksheets("tblBSTAUS").Cells(6, 2).Value = Me.txtBesch2.Text
        Worksheets("tblBSTAUS").Cells(7, 2).Value = Me.txtBesch3.Text
        Worksheets("tblBSTAUS").Cells(8, 2).Value = Me.txtBesch4.Text
        Worksheets("tblBSTAUS").Cells(9, 2).Value = Me.txtBesch5.Text
        Worksheets("tblBSTAUS").Cells(10, 2).Value = Me.txtBesch6.Text
        Worksheets("tblBSTAUS").Shapes.SelectAll
        Selection.Delete
        Worksheets("tblBSTAUS").Range("C5").Select
        Set varBild = Worksheets("tblBSTAUS").Pictures.Insert(varWelcherPath & Me.cboBsnr.Text
& ".jpg")
        Maskeloeschen
        Einfuegen
        Rem Selection. varBild
    End If
```

```
Private Sub cmdBeenden_Click()
    ActiveWindow.WindowState = xlNormal
    Me.Hide
End Sub
```

A1.7) Wie schon im Abschnitt A1.2.6 erläutert wurde, muss das Programm als Ergebnis Abb. A1.7a für den Fall „Ausgeben“ liefern.

Abb. A1.7a Das Layout beim Ausgeben



A1.8) **LITERATUR**

- Automatisierung und Programmierung mit Excel 2010 (www.rrzn.uni-hannover.de/buecher.html)
- VBA – Programmierung mit Excel von Johannes Gogolok (VBA mit Excel von Uni-Hagen.pdf)

A2) Aufgabe: Eine Datenbank für die Aufzinsung der Kapitalanlagen

Beschreibung

Der Zins, als Kompensation für den zwischenzeitlichen Konsumverzicht des Kapitalgebers, stellt eine von vier Kategorien der Zinsrechnung dar. Er ergibt sich in der ersten Zinsperiode aus der Multiplikation des vereinbarten Zinssatzes i mit dem Anfangskapital K0. Weiterhin steht das Endkapital Kn im Interesse des Gläubigers bzw. Schuldners, welches wiederrum von der Laufzeit n des Finanzkontraktes abhängig ist. Der Zinssatz i wird hierbei auch als Nominalzins bezeichnet. Daraus lassen sich die Formeln für das Endkapital bei einfacher Verzinsung und bei Zinseszinsrechnung ermitteln.

Bei dieser Aufgabe sollen die Datensätze für die `Person`, die `Kapitalanlage`, das `Anfangskapital` und den `Zinssatz in Prozent` über eine Userform in der Stammdatentabelle verwaltet werden. Mithilfe der Formel für die Zinsrechungen (Abb. A2.5a) und der Angabe von der Laufzeit sollen die verfügbaren und berechneten Daten in die Ausgabetabelle übertragen werden.

Wichtige Funktionen

Aufbau einer Exceldatei zu einem automatisierten Zinsrechnung mit folgenden Funktionen:

- Umgang mit den Zellenformaten und Wertintervallen
- Programmierung und Umgang mit den Optionsfeldern
- Konzeption und Realisation der Algorithmen zu der Aufgabestellung
- Umgang mit den Diagrammen

Vorgehensweise

Erstellen Sie eine Exceldatei mit dem Namen axxxxxxx.xLsm (xxxxxxx steht für Ihre Matrikelnummer). In dieser Aufgabe werden zwei Tabellen (`tblZinsEin` als Stammdatentabelle und `tblZinsAus` als Ausgabetabelle), eine Userform mit dem Namen `frmZins` und ein Diagrammblatt zur grafischen Darstellung `DZinsAus` benötigt.

A2.1) Das Tabellenblatt (Tab. A2.1a) kann `tblZinsEin` heißen und beinhaltet Datensätze für die `Person` (Spalte A), die `Kapitalanlage` (Spalte B), das `Anfangskapital` (Spalte C) und den `Zinssatz` (Spalte D).

	A	B	C	D
1	Frau Meyer	Deutsche Anleihen	120.000,00 €	2%
2	Herr Bandegani	Kreditvergabe	60.000,00 €	6,50%
3	Herr Müler	Immobilien	150.000,00 €	3,20%
4	Herr Schmidt	Aktien	140.000,00 €	8%
5				
6	**Person**	**Kapitalanlage**	**Anfangskapital**	**Zinssatz**

Tab. A2.1a `tblZinsEin`

A2.2) Das Userformblatt (Abb. 2.2a) kann `frmZins` heißen und besteht aus den folgenden Elementen:

Pos.	Bezeichnung (Caption)	Steuerelement	Name
1	Person:	Beschriftungsfeld	Nicht ändern

2	keine	Kombinationsfeld	cboPerson
3	Kapitalanlage:	Beschriftungsfeld	Nicht ändern
4	keine	Textfeld	txtAnlage
5	Anfangskapital:	Beschriftungsfeld	Nicht ändern
6	keine	Textfeld	txtKapital
7	Prozent	Beschriftungsfeld	Nicht ändern
8	keine	Textfeld	txtProzent
9	Zusatzdaten	Rahmen	Nicht ändern
10	Laufzeit im Jahr:	Beschriftungsfeld	Nicht ändern
11	keine	Kombinationsfeld	cboLaufzeit
12	Einfache Zinsrechnung:	Optionsfeld	optEZins
13	Zinsenzinsrechnung:	Optionsfeld	optZZins
14	Ablegen	Befehlsschaltfläche	cmdAblegen
15	Löschen	Befehlsschaltfläche	cmdloeschen
16	Ausgeben	Befehlsschaltfläche	cmdAusgeben
17	Beenden	Befehlsschaltfläche	cmdBeenden
18	Kennlinie…	Befehlsschaltfläche	cmdkennlinien
19	Formeln…	Befehlsschaltfläche	cmdInfo

Erläuterung zu dem Programmablauf

A2.2.1) Beim Starten des Userformblatts (Abb. 2.2a) müssen alle Einträge der ersten Spalte von der Tabelle (Tab. A2.1a) in das Kombinationsfeld `cboPerson` eingefügt werden.

A2.2.2) Wenn ein Eintrag für die Person gewählt wird, muss der entsprechende Datensatz aus der Tabelle (Tab. A2.1a) in den Textfeldern erscheinen.

A2.2.3) Wenn der Benutzer einen neuen Eintrag in das Userformblatt einträgt und auf „Ablegen“ klickt, muss der neue Datensatz in die Tabelle (Tab. A2.1a) hinzugefügt und anschließend sortiert werden.

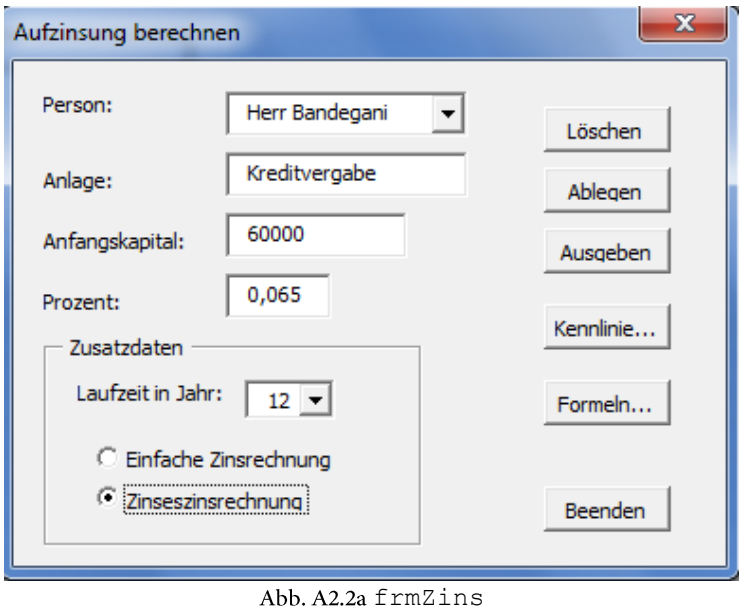


Abb. A2.2a `frmZins`

A2.2.4) Wenn der Benutzer einen Eintrag wählt, der schon in der Tabelle (Tab. A2.1a) existiert, muss der Datensatz beim Klicken auf „Ablegen“ ersetzt werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, dass der Datensatz ersetzt werden soll.)

A2.2.5) Wenn der Benutzer einen Eintrag wählt und auf die Befehlsschaltfläche „Löschen“ klickt, muss der Datensatz aus der Tabelle (Tab. A2.1a) gelöscht werden. (Hierbei soll eine

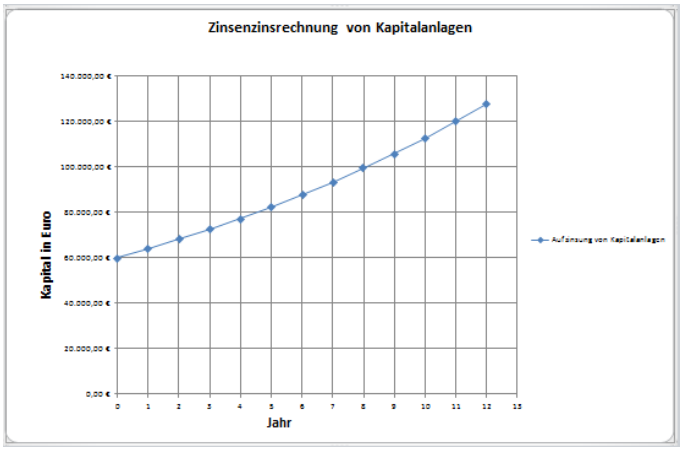
Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, dass der Datensatz gelöscht werden soll.)
A2.2.6) Wenn der Benutzer auf die Befehlsschaltfläche „Ausgeben“ klickt, müssen alle in der Userform (Abb. A2.2a) augewälten Daten mit den berechneten Werten für den Kapitalwert (Abb. A2.5a) in die Ausgabetabelle (Tab. A2.3a) übertragen werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, wenn der Benutzer keinen Eintrag gewählt hat.)
A2.2.7) Die Befehlsschaltfläche „Beenden“ blendet die Userform aus.

2.1. Erstellen Sie eine Tabelle (Tab. A2.3a) mit dem Namen `tblZinsAus`. Die Werte für die `Person` (Zeile 2/Spalte B), die `Kapitalanlage` (Zeile 3/Spalte B), das `Anfangskapital` (Zeile 4/Spalte B), den `Prozentsatz` (Zeile 5/Spalte B), die `Laufzeit im Jahr` (Zeile 6 /Spalte B) müssen von der Userform (Abb. A2.2a) in diese Tabelle übertragen werden. Die zu der Laufzeit im Jahr errechneten Kapitalwerte (Abb. A2.5a) müssen ab Zeile 9 in die Spalte B platziert werden.

2.2. Erstellen Sie ein Diagrammblatt (Tab. A2.4a) und bennen Sie es mit dem Namen `DZinsAus` (Tab. A2.4a). Dabei beziehen die x-Achse auf das Jahr (Spalte A / Zeilen 9 bis 29) und die y-Achse auf den Kapitalwert (Spalte B / Zeilen 9 bis 29) von der Tabelle `tblZinsAus`(Tab. A2.3a).

	A	B	C
1	Aufzinsung für die Kapitalanlagen		
2	**Investmentperson:**	Herr BandegaNl	
3	**Kapitalanlage:**	Kreditvergabe	
4	**Anfangskapital:**	60.000,00 €	
5	**Prozentsatz:**	0,065	
6	**Laufzeit im Jahr:**	12	
7			
8	**Jahr**	**Kapitalwert**	
9	0	60.000,00 €	
10	1	63.900,00 €	
11	2	68.053,50 €	
12	3	72.476,98 €	
13	4	77.187,98 €	
14	5	82.205,20 €	
15	6	87.548,54 €	
16	7	93.239,19 €	
17	8	99.299,74 €	
18	9	105.754,22 €	
19	10	112.628,25 €	
20	11	119.949,08 €	
21	12	127.745,77 €	
22			
23			

Tab. A2.3a `tblZinsAus`



Tab. A2.4a `DZinsAus`

Abb. A2.5a **`Formeln zur Verzinsung`**

1) Einfache Verzinsung: $K_n = K_0(1 + i \,.\, n)$

2) Zinseszinsrechnung: $K_n = K_0(1 + i)^n$

Programmablaufplan

Der Programmablaufplan (Abb. A2.6a) bezieht sich auf die Abschnitte A2.2.1 – A2.2.7 Entwickeln und realisieren Sie bitte einen Programmablaufplan, in dem kurze Codes und vereinfachte Aktionen für den Benutzer ermöglicht werden.

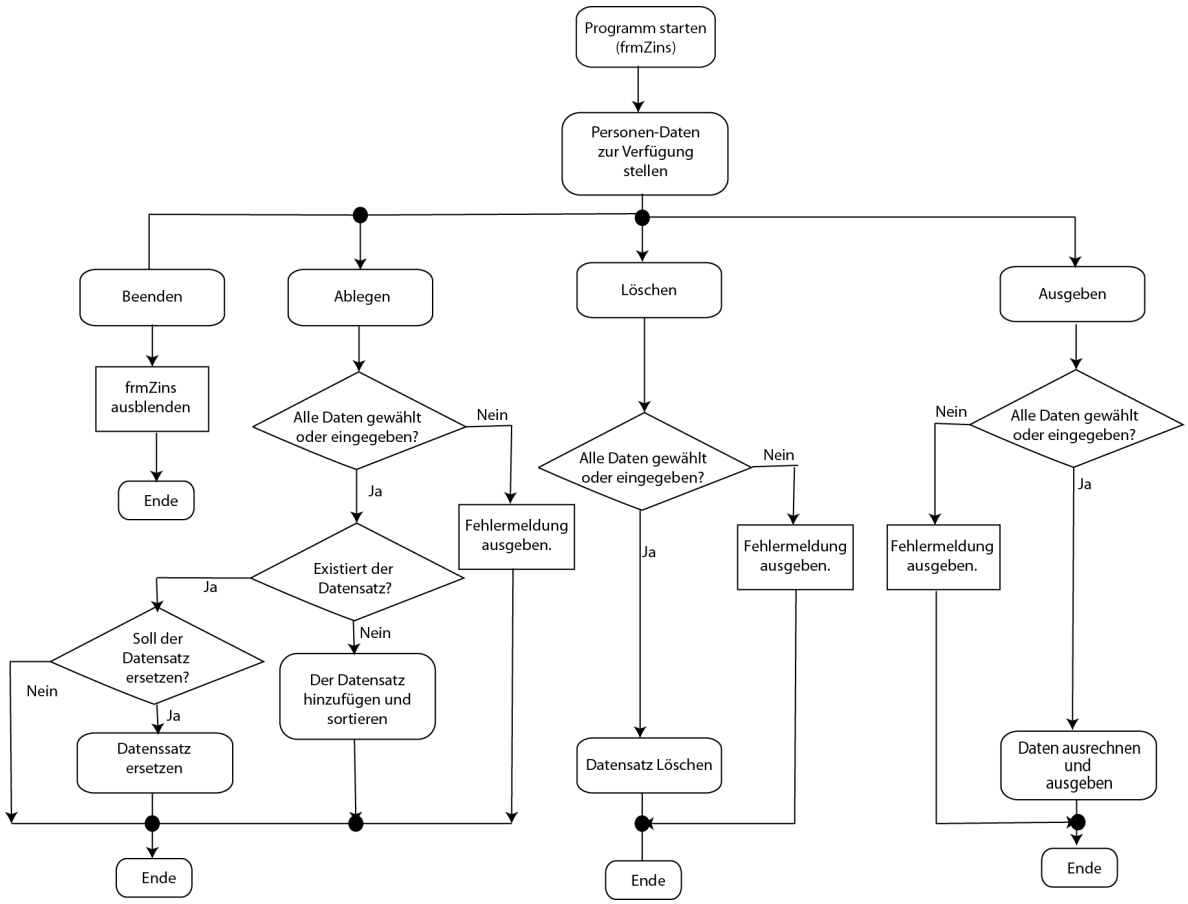


Abb. A2.6a Programmablaufplan zu der Verzinsung

A2.6) Folgende VBA-Codes dienen dazu, in den Abschnitten A2.2.1-A2.2.7 beschriebenen Programmablaufplan zu implementieren:

```
Dim varWelcherEintrag As Integer
Private Sub UserForm_Initialize()
   Me.optEZins.Value = True
   Me.optZZins.Value = False
   Worksheets("tblZinsEin").Activate
   Maskeloeschen
   Einfuegen "tblZinsEin", Me.cboPerson, 1
   Me.cboLaufzeit.AddItem 5
   Me.cboLaufzeit.AddItem 8
   Me.cboLaufzeit.AddItem 10
   Me.cboLaufzeit.AddItem 12
   Me.cboLaufzeit.AddItem 15
   Me.cboLaufzeit.AddItem 20
End Sub
Sub Einfuegen(Tabellenname As String, KomboBox As Object, Spaltennummer As Integer)
   Dim varEndwert As Integer
   Dim i As Integer
   varEndwert = Worksheets(Tabellenname).Cells(1, 1).CurrentRegion.Rows.Count
   i = 0
   For i = 1 To varEndwert
      KomboBox.AddItem Worksheets(Tabellenname).Cells(i, Spaltennummer).Value
```

```
    Next i
End Sub
Private Sub cboPerson_Click()
    varWelcherEintrag = Me.cboPerson.ListIndex + 1
    Me.txtAnlage.Text = Worksheets("tblZinsEin").Cells(varWelcherEintrag, 2).Value
    Me.txtKapital.Text = Worksheets("tblZinsEin").Cells(varWelcherEintrag, 3).Value
    Me.txtProzent.Text = Worksheets("tblZinsEin").Cells(varWelcherEintrag, 4).Value
End Sub
Private Sub cboPerson_Enter()
    Me.txtAnlage.Text = ""
    Me.txtKapital.Text = ""
    Me.txtProzent.Text = ""
    Me.cboLaufzeit.Text = ""
    Me.optEZins.Value = True
    Me.optZZins.Value = False
End Sub
Private Sub cboPerson_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Me.txtAnlage.Text = ""
    Me.txtKapital.Text = ""
    Me.txtProzent.Text = ""
    Me.cboLaufzeit.Text = ""
    Me.optEZins.Value = True
    Me.optZZins.Value = False
End Sub
Private Sub cmdloeschen_Click()
 If Me.cboPerson.Text = "" Then
       MsgBox "Sie haben keinen Eintrag gewählt"
       Beep
       Exit Sub
    Else
       Meldung = "Wollen Sie wirklich den Datensatz löschen?"
       Titel = "Datensatz löschen"
       Stil = vbYesNo + vbCritical + vbDefaultButton2
       Ergebnis = MsgBox(Meldung, Stil, Titel)
       If Ergebnis = vbYes Then
          Worksheets("tblZinsEin").Rows(varWelcherEintrag).Delete Shift:=x1up
          Maskeloeschen
          Einfuegen "tblZinsEin", Me.cboPerson, 1
       Else
          Exit Sub
       End If
    End If
End Sub
Sub Maskeloeschen()
    Me.cboPerson.Clear
    Me.cboPerson.Text = ""
    Me.txtAnlage.Text = ""
    Me.txtKapital.Text = ""
    Me.txtProzent.Text = ""
    Me.cboLaufzeit.Text = ""
End Sub
Private Sub cmdAblegen_Click()
    If Me.cboPerson.Text = "" Then
       MsgBox "Sie müssen einen Eintrag wählen!"
       Beep
       Exit Sub
    Else
       If varWelcherEintrag > 0 And Me.cboPerson.Text > "" Then
          If Worksheets("tblZinsEin").Cells(varWelcherEintrag, 1).Value = Me.cboPerson.Text Then
             Dim Meldung As String
             Dim Titel As String
             Dim Stil As Integer
             Dim Ergebnis As Integer
             Stil = vbYesNo + vbCritical + vbDefaultButton2
             Meldung = "Wollen Sie den Datensatz ersetzen?"
```

```
            Titel = "Datensatz ersetzen"
            Ergebnis = MsgBox(Meldung, Stil, Titel)
            If Ergebnis = vbYes Then
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 2).NumberFormat = "@"
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 2).Value = Me.txtAnlage.Text
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 3).NumberFormat = "#,##0.00 $"
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 3).Value = Me.txtKapital.Text
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 4).NumberFormat = "0.00%"
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 4).Value = Me.txtProzent.Text
               Maskeloeschen
               Einfuegen "tblZinsEin", Me.cboPerson, 1
               Worksheets("tblZinsEin").Activate
            Else
               Exit Sub
            End If

         Else
            Neueintragen
            Maskeloeschen
            Einfuegen "tblZinsEin", Me.cboPerson, 1
            Worksheets("tblZinsEin").Activate
         End If
      Else
         Neueintragen
         Maskeloeschen
         Einfuegen "tblZinsEin", Me.cboPerson, 1
         Worksheets("tblZinsEin").Activate
      End If
   End If
End Sub
```

```
Sub Neueintragen()
   Dim VarNeueZeile As Integer
   VarNeueZeile = Worksheets("tblZinsEin").Cells(1, 1).CurrentRegion.Rows.Count + 1
   Worksheets("tblZinsEin").Cells(VarNeueZeile, 1).NumberFormat = "@"
   Worksheets("tblZinsEin").Cells(VarNeueZeile, 1).Value = Me.cboPerson.Text
   Worksheets("tblZinsEin").Cells(VarNeueZeile, 2).NumberFormat = "@"
   Worksheets("tblZinsEin").Cells(VarNeueZeile, 2).Value = Me.txtAnlage.Text
   Worksheets("tblZinsEin").Cells(VarNeueZeile, 3).NumberFormat = "#,##0.00 $"
   Worksheets("tblZinsEin").Cells(VarNeueZeile, 3).Value = Me.txtKapital.Text
   Worksheets("tblZinsEin").Cells(VarNeueZeile, 4).NumberFormat = "0.00%"
   Worksheets("tblZinsEin").Cells(VarNeueZeile, 4).Value = Me.txtProzent.Text
   With Worksheets("tblZinsEin")
      .Range(.Cells(1, 1), .Cells(VarNeueZeile, 4)).Sort Key1:= _
      Worksheets("tblZinsEin").Columns("A")
   End With
   varWelcherEintrag = VarNeueZeile
End Sub
```

```
Private Sub cmdAusgeben_Click()
   If Me.cboPerson.Text > "" And Me.txtAnlage.Text > "" And Me.txtKapital.Text > "" _
      And Me.txtProzent.Text > "" And Me.cboLaufzeit.Text > "" Then
      Worksheets("tblZinsAus").Activate
      Ausgabeloeschen
      Worksheets("tblZinsAus").Cells(2, 1).NumberFormat = "@"
      Worksheets("tblZinsAus").Cells(2, 1).Value = "Investmentperson:"
      Worksheets("tblZinsAus").Cells(2, 2).NumberFormat = "@"
      Worksheets("tblZinsAus").Cells(2, 2).Value = Me.cboPerson.Text
      Worksheets("tblZinsAus").Cells(3, 1).NumberFormat = "@"
      Worksheets("tblZinsAus").Cells(3, 1).Value = "Kapitalanlage:"
      Worksheets("tblZinsAus").Cells(3, 2).NumberFormat = "@"
      Worksheets("tblZinsAus").Cells(3, 2).Value = Me.txtAnlage.Text
      Worksheets("tblZinsAus").Cells(4, 1).NumberFormat = "@"
      Worksheets("tblZinsAus").Cells(4, 1).Value = "Anfangskapital:"
      Worksheets("tblZinsAus").Cells(4, 2).NumberFormat = "#,##0.00 $"
      Worksheets("tblZinsAus").Cells(4, 2).Value = Me.txtKapital.Text
      Worksheets("tblZinsAus").Cells(5, 1).NumberFormat = "@"
      Worksheets("tblZinsAus").Cells(5, 1).Value = "Prozentsatz:"
```

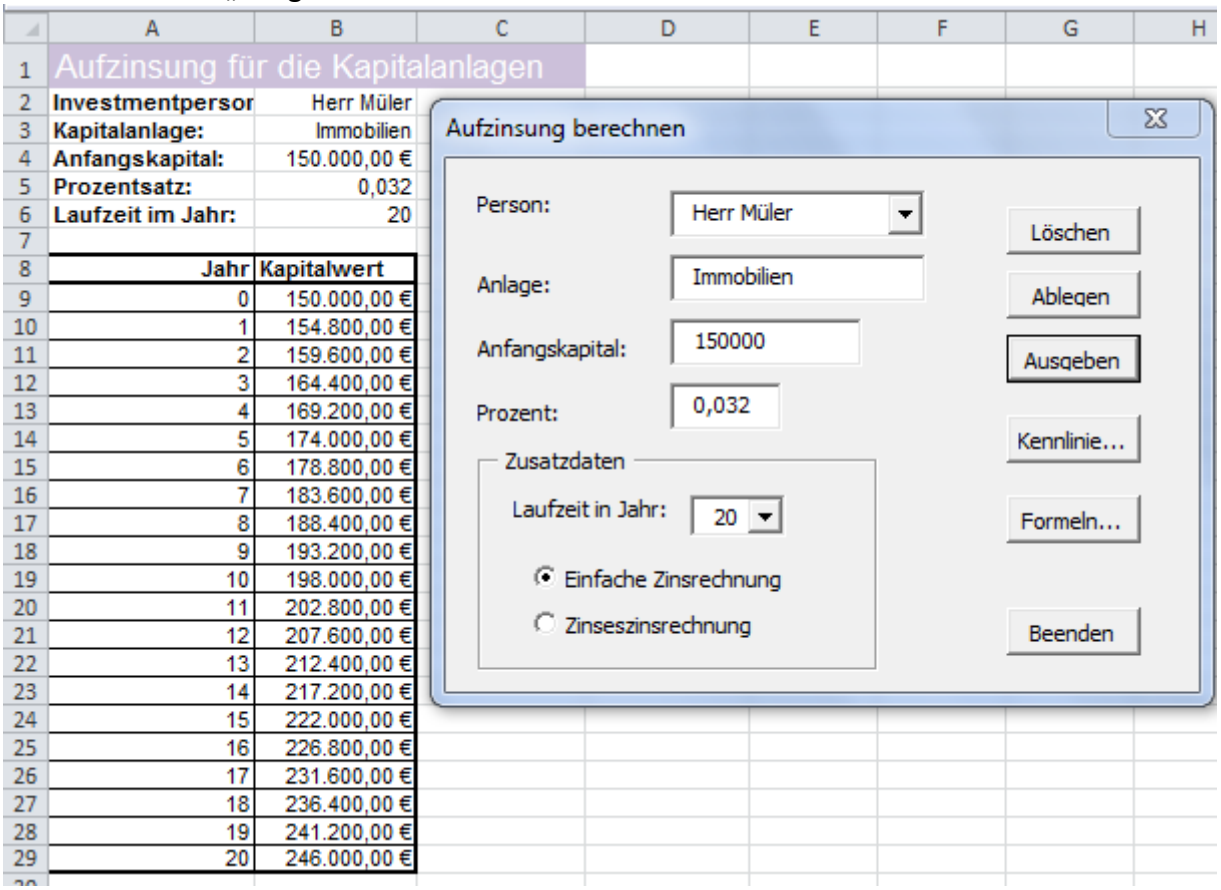
```
        Worksheets("tblZinsAus").Cells(5, 2).NumberFormat = "0.00%"
        Worksheets("tblZinsAus").Cells(5, 2).Value = Me.txtProzent.Text
        Worksheets("tblZinsAus").Cells(6, 2).NumberFormat = "0"
        Worksheets("tblZinsAus").Cells(6, 2).Value = Me.cboLaufzeit.Text
        Worksheets("tblZinsAus").Cells(8, 1).NumberFormat = "@"
        Worksheets("tblZinsAus").Cells(8, 1).Value = "Jahr"
        Worksheets("tblZinsAus").Cells(8, 2).NumberFormat = "@"
        Worksheets("tblZinsAus").Cells(8, 2).Value = "Kapitalwert"
        If Me.optEZins.Value = True Then
            i = 0
            For i = 0 To Me.cboLaufzeit.Text
                Worksheets("tblZinsAus").Cells(9 + i, 1).Value = i
                Worksheets("tblZinsAus").Cells(9 + i, 2).NumberFormat = "#,##0.00 $"
                Worksheets("tblZinsAus").Cells(9 + i, 2).Value = Me.txtKapital.Text * (1 + Me.txtProzent.Text * i)
            Next i
        ElseIf Me.optZZins.Value = True Then
            i = 0
            For i = 0 To Me.cboLaufzeit.Text
                Worksheets("tblZinsAus").Cells(9 + i, 1).Value = i
                Worksheets("tblZinsAus").Cells(9 + i, 2).NumberFormat = "#,##0.00 $"
                Worksheets("tblZinsAus").Cells(9 + i, 2).Value = Me.txtKapital.Text * (1 + Me.txtProzent.Text * 1) ^ i
            Next i
        Else
            MsgBox "Ein Fehler bei der Laufzeit ist passiert."
        End If
    Else
        MsgBox "Sie müssen einen Eintrag wählen, oder ein Feld ist leer!"
    End If
End Sub
```

```
Sub Ausgabeloeschen()
    For i = 0 To 4
        Worksheets("tblZinsAus").Cells(2 + i, 2).Value = Null
    Next i
    For i = 0 To 20
        Worksheets("tblZinsAus").Cells(i + 9, 1).Value = Null
        Worksheets("tblZinsAus").Cells(i + 9, 2).Value = Null
    Next i
End Sub
```

```
Private Sub cmdInfo_Click()
    Worksheets("tblZinsInfo").Activate
End Sub
```

```
Private Sub cmdBeenden_Click()
    Me.Hide
End Sub
```

A2.7) Wie schon im Abschnitt A2.2.6 erläutert wurde, muss das Programm als Ergebnis Abb. A2.7a für den Fall „Ausgeben“ liefern.



	A	B	C	D	E	F	G	H
1	Aufzinsung für die Kapitalanlagen							
2	Investmentpersor	Herr Müler						
3	Kapitalanlage:	Immobilien						
4	Anfangskapital:	150.000,00 €						
5	Prozentsatz:	0,032						
6	Laufzeit im Jahr:	20						
7								
8	Jahr	Kapitalwert						
9	0	150.000,00 €						
10	1	154.800,00 €						
11	2	159.600,00 €						
12	3	164.400,00 €						
13	4	169.200,00 €						
14	5	174.000,00 €						
15	6	178.800,00 €						
16	7	183.600,00 €						
17	8	188.400,00 €						
18	9	193.200,00 €						
19	10	198.000,00 €						
20	11	202.800,00 €						
21	12	207.600,00 €						
22	13	212.400,00 €						
23	14	217.200,00 €						
24	15	222.000,00 €						
25	16	226.800,00 €						
26	17	231.600,00 €						
27	18	236.400,00 €						
28	19	241.200,00 €						
29	20	246.000,00 €						

Abb. A2.7a Das Layout beim Ausgeben

A2.8) **LITERATUR**

- Investition und Finanzierung von Ulrich Ermschel et al., ISBN-978-3-7908-2745-3, Springer-Verlag Berlin Heidelberg 2011(https://kataloge.uni-hamburg.de/DB=2/SET=5/TTL=1/SHW?FRST=3)
- Automatisierung und Programmierung mit Excel 2010 (www.rrzn.uni-hannover.de/buecher.html)
- VBA – Programmierung mit Excel von Johannes Gogolok (VBA mit Excel von Uni-Hagen.pdf)

A3) Aufgabe: Eine Datenbank für die Tilgungsrechnung

Beschreibung

Erstellen Sie eine Datenbank, mit der das prinzipielle Verhalten einer Tilgungsrechnung realisiert werden kann. Aktuell bietet eine Bank vier verschiedene Kredite mit entsprechendem risikobehafteten Zinssatz. In dieser Aufgabe müssen die Berechnungen der Restschuld, der Zinszahlung, der Tilgung und der Annuität in Abhängigkeit von der Kredithöhe, der Laufzeit und der Zinshöhe (Anleihekredit mit 2%, Hypothekenkredit mit 3,4%, Policenkredit mit 4,89% und Unternehmenskredit mit 7,3%) erreicht werden. Die ausgewählten Daten (Firma, Anschrift, Postleitzahl, Ort, Kredithöhe, Laufzeit und Kreditart) müssen mit den **berechneten** Daten (Abb. A3.5a) in der Ausgabetabelle (Tab. A3.3a) erscheinen. Anschließend soll die **grafische** Darstellung der Zinszahlung und der Annuität in Abhängigkeit von der laufzeit im Jahr angezeigt werden. (Tab. A3.4a)

Wichtige Funktionen

Aufbau einer Exceldatei zu einer automatisierten Zinsrechnung mit folgenden Funktionen:

- Umgang mit den Zellenformaten und Wertintervallen
- Programmierung und Umgang mit den Optionsfeldern
- Konzeption und Realisation der Algorithmen zu der Aufgabestellung
- Umgang mit den Diagrammen

Vorgehensweise

Erstellen Sie eine Exceldatei mit dem Namen axxxxxxx.xLsm (xxxxxxx steht für Ihre Matrikelnummer). In dieser Aufgabe werden zwei Tabellen (`tblTilgungEin` als Stammdatentabelle und `tblTilgungAus` als Ausgabetabelle), eine Userform mit dem Namen `frmTilgung` und ein Diagrammblatt zur grafischen Darstellung `DTilgungAus` benötigt.
A3.1) Das Tabellenblatt (Tab. A3.1a) kann `tblTilgungEin` heißen und beinhaltet Datensätze für die `Firma` (Spalte A), die `Anschrift` (Spalte B), die `Postleitzahl` (Spalte C) und den `Ort` (Spalte D).

	A	B	C	D
1	H&M AG	Postfach 2318	D-36243	Niederaula
2	C&A	Postfach 101111	D-40002	Düsseldorf
3	Hetex Jacquard AG	Im Bifang 7	CH-5102	Rupperswil
4	IFM Immobilien AG	Karl-Ludwig-Straße 2	D-69117	69117 Heidelberg
5				
6				
7	**Firma**	**Anschrift**	**Postleitzahl**	**Ort**

Tab. A3.1a `tblZinsEin`

A3.2) Das Userformblatt (Abb. A3.2a) kann `frmTilgung` heißen und besteht aus den folgenden Elementen:

Pos.	Bezeichnung (Caption)	Steuerelement	Name
1	Firma:	Beschriftungsfeld	Nicht ändern
2	Keine	Kombinationsfeld	cboFirma
3	Anschrift:	Beschriftungsfeld	Nicht ändern

4	Keine	Textfeld	txtAnschrift
5	Postleitzahl:	Beschriftungsfeld	Nicht ändern
6	keine	Textfeld	txtPLZ
7	Ort:	Beschriftungsfeld	Nicht ändern
8	keine	Textfeld	txtOrt
9	Zusatzdaten	Rahmen	Nicht ändern
10	Kredithöhe:	Beschriftungsfeld	Nicht ändern
11	keine	Textfeld	txtKredit
12	Laufzeit im Jahr:	Beschriftungsfeld	Nicht ändern
13	keine	Kombinationsfeld	cboLaufzeit
14	Einleihenkredit 2%	Optionsfeld	optANK
15	Policenkredit 4,89%	Optionsfeld	optPLK
16	Hypothekenkredit 3,4%	Optionsfeld	optHYK
17	Unternehmenskredit 7,3%	Optionsfeld	optUNK
18	Ablegen	Befehlsschaltfläche	cmdAblegen
19	Löschen	Befehlsschaltfläche	cmdloeschen
20	Ausgeben	Befehlsschaltfläche	cmdAusgeben
21	Beenden	Befehlsschaltfläche	cmdBeenden
22	Kennlinie…	Befehlsschaltfläche	cmdkennlinien
23	Formeln…	Befehlsschaltfläche	cmdInfo

Erläuterung zu dem Programmablauf

A3.2.1) Beim Starten des Userformblatts (Abb. A3.2a) müssen alle Einträge der ersten Spalte von der Tabelle `tblTilgungEin` (Tab. A3.1a) in das Kombinationsfeld `cboFirma` eingefügt werden.

A3.2.2) Wenn ein Eintrag für die Person gewählt wird, muss der entsprechende Datensatz aus der Tabelle `tblTilgungEin` in den Textfeldern erscheinen.

A3.2.3) Wenn der Benutzer einen neuen Eintrag in das Userformblatt einträgt und auf „Ablegen“ klickt, muss der neue Datensatz in die Tabelle `tblTilgungEin` hinzugefügt werden und anschließend sortiert werden.

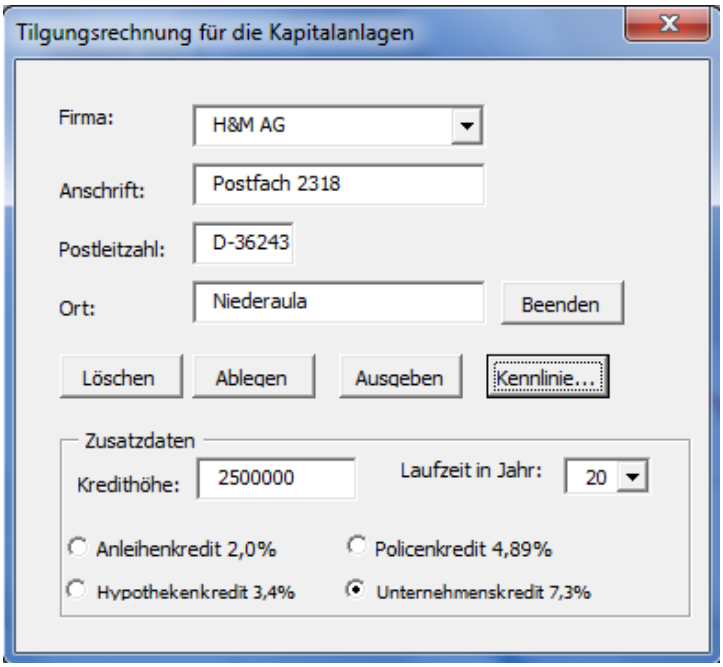


Abb. A3.2a `frmTilgung`

A3.2.4) Wenn der Benutzer einen Eintrag wählt, der schon in der Tabelle `tblTilgungEin` existiert, muss der Datensatz beim Klicken auf „Ablegen“ ersetzt werden. (Hierbei soll eine

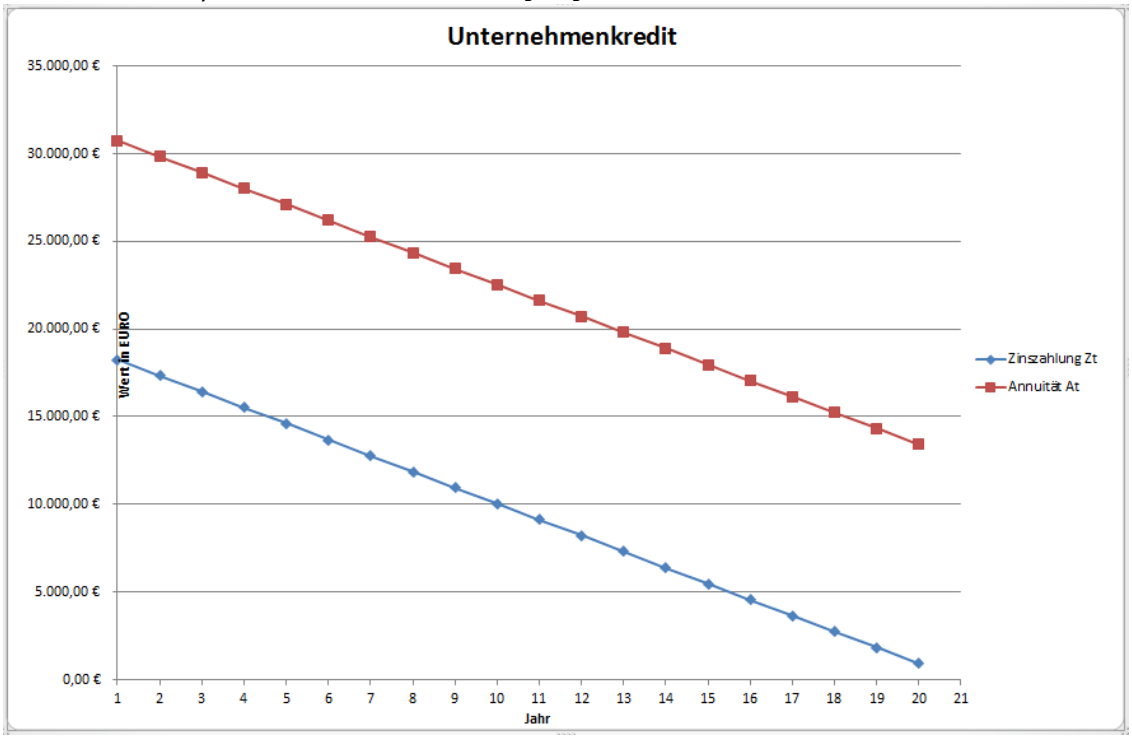
Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz ersetzt werden soll.)
A3.2.5) Wenn der Benutzer einen Eintrag wählt und auf die Befehlsschaltfläche „Löschen“ klickt, muss der Datensatz aus der Tabelle `tblTilgungEin` gelöscht werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz gelöscht werden soll.)
A3.2.6) Wenn der Benutzer auf die Befehlsschaltfläche „Ausgeben“ klickt, müssen alle in der Userform `frmTilgung` augewälten Daten mit den berechneten Werten für die Restschuld, Zinszahlung, Tilgung und die Annuität (Abb. A2.5a) in die Ausgabetabelle(Tab. A3.3a) `tblTilgungAus` übertragen werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, wenn der Benutzer keinen Eintrag gewählt hat.)
A3.2.7) Die Befehlsschaltfläche „Beenden“ blendet die Userform aus.

A3.3) Erstellen Sie eine Tabelle(Tab. A3.3a) mit dem Namen `tblTilgungAus`. Die Werte für die `Firma` (Zeile 2/Spalte B), `Anschrift` (Zeile 3/Spalte B), `Postleitzahl`, den `Ort` (Zeile 4/Spalte B), die `Kredithöhe` (Zeile 5/Spalte B), `Kreditart` (Zeile 6 /Spalte B), den `Zinssatz` (Zeile 7 /Spalte B) und die `Laufzeit im Jahr` (Zeile 8 /Spalte B) müssen von der Userform `frmTilgung` und die zu der Laufzeit im Jahr errechneten `Restschuld`(Spalte B), `Zinszahlung`(Spalte C), `Tilgung`(Spalte D) und `Annuität`(Spalte E) müssen ab Zeile 11 in die Tabelle(Tab. A3.3a) übertragen werden.

	A	B	C	D	E
1	Tilgungsrechnung für die Kapitalanlagen				
2	**Firma:**	H&M AG			
3	**Anschrift:**	Postfach 2318			
4	**PLZ + Ort:**	D-36243 Niederaula			
5	**Kredithöhe:**	250.000,00 €			
6	**Kreditart:**	Unternehmenskredit			
7	**Zinssatz:**	7,300%			
8	**Laufzeit im Jahr**	20 Jahr			
9					
10	**Jahr**	**Restschuld Kt**	**Zinszahlung Zt**	**Tilgung Tt**	**Annuität At**
11	1	250.000,00 €	18.250,00 €	12.500,00 €	30.750,00 €
12	2	237.500,00 €	17.337,50 €	12.500,00 €	29.837,50 €
13	3	225.000,00 €	16.425,00 €	12.500,00 €	28.925,00 €
14	4	212.500,00 €	15.512,50 €	12.500,00 €	28.012,50 €
15	5	200.000,00 €	14.600,00 €	12.500,00 €	27.100,00 €
16	6	187.500,00 €	13.687,50 €	12.500,00 €	26.187,50 €
17	7	175.000,00 €	12.775,00 €	12.500,00 €	25.275,00 €
18	8	162.500,00 €	11.862,50 €	12.500,00 €	24.362,50 €
19	9	150.000,00 €	10.950,00 €	12.500,00 €	23.450,00 €
20	10	137.500,00 €	10.037,50 €	12.500,00 €	22.537,50 €
21	11	125.000,00 €	9.125,00 €	12.500,00 €	21.625,00 €
22	12	112.500,00 €	8.212,50 €	12.500,00 €	20.712,50 €
23	13	100.000,00 €	7.300,00 €	12.500,00 €	19.800,00 €
24	14	87.500,00 €	6.387,50 €	12.500,00 €	18.887,50 €
25	15	75.000,00 €	5.475,00 €	12.500,00 €	17.975,00 €
26	16	62.500,00 €	4.562,50 €	12.500,00 €	17.062,50 €
27	17	50.000,00 €	3.650,00 €	12.500,00 €	16.150,00 €
28	18	37.500,00 €	2.737,50 €	12.500,00 €	15.237,50 €
29	19	25.000,00 €	1.825,00 €	12.500,00 €	14.325,00 €
30	20	12.500,00 €	912,50 €	12.500,00 €	13.412,50 €

Tab. A3.3a `tblTilgungAus`

A3.4) Erstellen Sie ein Diagrammblatt (Tab. A3.4a) und bennen Sie es mit dem Namen `DTilgungAus`. Dabei beziehen Sie die x-Achse auf das Jahr (Spalte A / Zeilen 11 bis 30) und die y-Achse auf die Werte der Zinszahlung(Spalte C /Zeilen 11 bis 30) und der Annuität (Spalte E / Zeilen 11 bis 30) von der Tabelle `tblTilgungAus`.



Tab. A3.4a `DTilgungAus`

Abb. A3.5a**) Formeln und Erklärung:**

Tilgungsrechnung

Ziel der Tilgungsrechnung ist die Aufstellung eines vollständigen Tilgungsplanes, in dem etwas über den zeitlichen Verlauf der Restschuld sowie der Zins- und Tilgungszahlung steht. Dabei wird wie folgt aufgestellt:

- Restschuld K_t im Zeitpunkt t (dabei ist K_0 die Kredithöhe) $\boxed{K_t = K_{n-1} - T_t}$ bei n >= 1, bei n=1 $\boxed{K_t = K_0}$
- Tilgungsrate T_t im Zeitpunkt t : $\boxed{T_t = \frac{K_0}{n}}$
- Zinszahlung Z_t Zeitpunkt t: $\boxed{Z_t = i * K_{n-1}}$
- Annuität (Belastung) A_t im Zeitpunkt t: $\boxed{A_t = Z_t + T_t}$
- Zinssatz i der Kreditschuld
- Laufzeit n der Kreditschuld

Programmablaufplan

Der Programmablaufplan (Abb. A3.6a) bezieht sich auf die Abschnitten A3.2.1 – A3.2.7

Entwickeln und realisieren Sie bitte einen Programmablaufplan und ein Programmcode, in dem kurze Codes und vereinfachte Aktionen für den Benutzer ermöglicht werden.

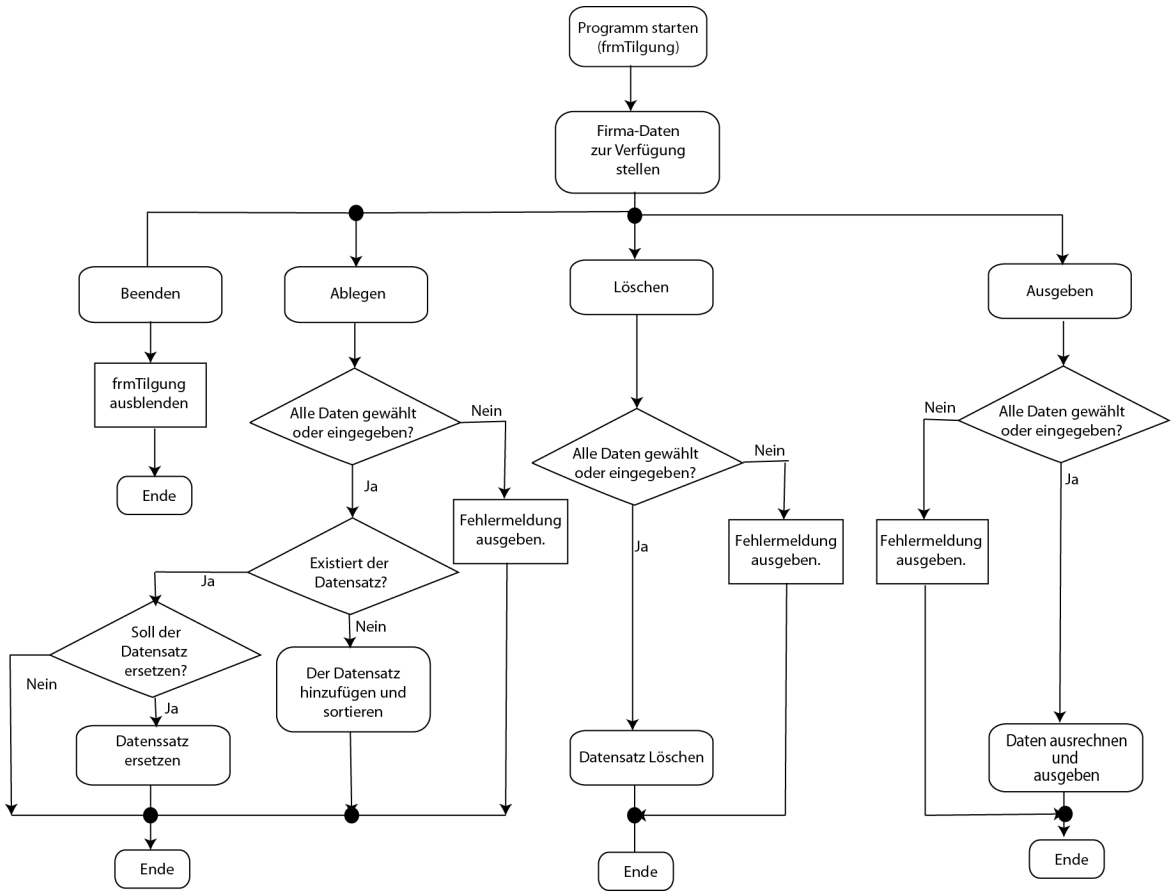


Abb. A3.6a Programmablaufplan zu der Tilgungsrechnung

4.3. Folgende VBA-Codes dienen dazu, um den in den Abschnitten A3.2.1-A3.2.7 beschriebenen Programmablaufplan zu implementieren:

```
Dim varWelcherEintrag As Integer
Private Sub UserForm_Initialize()
   Me.optANK.Value = True
   Me.optHYK.Value = False
   Me.optPLK.Value = False
   Me.optUNK.Value = False
   Worksheets("tblTilgungEin").Activate
   Maskeloeschen
   Einfuegen "tblTilgungEin", Me.cboFirma, 1
   Me.cboLaufzeit.AddItem 5
   Me.cboLaufzeit.AddItem 8
   Me.cboLaufzeit.AddItem 10
   Me.cboLaufzeit.AddItem 12
   Me.cboLaufzeit.AddItem 15
   Me.cboLaufzeit.AddItem 20
End Sub
```

```
Sub Einfuegen(Tabellenname As String, KomboBox As Object, Spaltennummer As Integer)
   Dim varEndwert As Integer
   Dim i As Integer
   varEndwert = Worksheets(Tabellenname).Cells(1, 1).CurrentRegion.Rows.Count
   i = 0
```

```
    For i = 1 To varEndwert
        KomboBox.AddItem Worksheets(Tabellenname).Cells(i, Spaltennummer).Value
    Next i
End Sub
Private Sub cboFirma_Click()
    varWelcherEintrag = Me.cboFirma.ListIndex + 1
    Me.txtAnschrift.Text = Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 2).Value
    Me.txtPLZ.Text = Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 3).Value
    Me.txtOrt.Text = Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 4).Value
End Sub
Private Sub cboFirma_Enter()
    Me.txtAnschrift.Text = ""
    Me.txtPLZ.Text = ""
    Me.txtOrt.Text = ""
    Me.cboLaufzeit.Text = ""
    Me.optANK.Value = True
    Me.optHYK.Value = False
    Me.optPLK.Value = False
    Me.optUNK.Value = False
End Sub
Private Sub cboFirma_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Me.txtAnschrift.Text = ""
    Me.txtPLZ.Text = ""
    Me.txtOrt.Text = ""
    Me.cboLaufzeit.Text = ""
    Me.optANK.Value = True
    Me.optHYK.Value = False
    Me.optPLK.Value = False
    Me.optUNK.Value = False
End Sub
Private Sub cmdloeschen_Click()
 If Me.cboFirma.Text = "" Then
        MsgBox "Sie haben keinen Eintrag gewählt"
        Beep
        Exit Sub
    Else
        Meldung = "Wollen Sie wirklich den Datensatz löschen?"
        Titel = "Datensatz löschen"
        Stil = vbYesNo + vbCritical + vbDefaultButton2
        Ergebnis = MsgBox(Meldung, Stil, Titel)
        If Ergebnis = vbYes Then
            Worksheets("tblTilgungEin").Rows(varWelcherEintrag).Delete Shift:=x1up
            Maskeloeschen
            Einfuegen "tblTilgungEin", Me.cboFirma, 1
        Else
            Exit Sub
        End If
    End If
End Sub
Sub Maskeloeschen()
    Me.cboFirma.Clear
    Me.cboFirma.Text = ""
    Me.txtAnschrift.Text = ""
    Me.txtPLZ.Text = ""
    Me.txtOrt.Text = ""
    Me.cboLaufzeit.Text = ""
End Sub
Private Sub cmdAblegen_Click()
    If Me.cboFirma.Text = "" Then
        MsgBox "Sie müssen einen Eintrag wählen!"
        Beep
        Exit Sub
    Else
        If varWelcherEintrag > 0 And Me.cboFirma.Text > "" Then
            If Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 1).Value = Me.cboFirma.Text Then
```

```
            Dim Meldung As String, Titel As String
            Dim Stil As Integer,  Ergebnis As Integer
            Stil = vbYesNo + vbCritical + vbDefaultButton2
            Meldung = "Wollen Sie den Datensatz ersetzen?"
            Titel = "Datensatz ersetzen"
            Ergebnis = MsgBox(Meldung, Stil, Titel)
            If Ergebnis = vbYes Then
               Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 2).NumberFormat = "@"
               Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 2).Value = Me.txtAnschrift.Text
               Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 3).NumberFormat = "@"
               Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 3).Value = Me.txtPLZ.Text
               Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 4).NumberFormat = "@"
               Worksheets("tblTilgungEin").Cells(varWelcherEintrag, 4).Value = Me.txtOrt.Text
               Maskeloeschen
               Einfuegen "tblTilgungEin", Me.cboFirma, 1
               Worksheets("tblTilgungEin").Activate
            Else
               Exit Sub
            End If
         Else
            Neueintragen
            Maskeloeschen
            Einfuegen "tblTilgungEin", Me.cboFirma, 1
            Worksheets("tblTilgungEin").Activate
         End If
      Else
         Neueintragen
         Maskeloeschen
         Einfuegen "tblTilgungEin", Me.cboFirma, 1
         Worksheets("tblTilgungEin").Activate
      End If
   End If
End Sub
```

```
Sub Neueintragen()
   Dim VarNeueZeile As Integer
   VarNeueZeile = Worksheets("tblTilgungEin").Cells(1, 1).CurrentRegion.Rows.Count + 1
   Worksheets("tblTilgungEin").Cells(VarNeueZeile, 1).NumberFormat = "@"
   Worksheets("tblTilgungEin").Cells(VarNeueZeile, 1).Value = Me.cboFirma.Text
   Worksheets("tblTilgungEin").Cells(VarNeueZeile, 2).NumberFormat = "@"
   Worksheets("tblTilgungEin").Cells(VarNeueZeile, 2).Value = Me.txtAnschrift.Text
   Worksheets("tblTilgungEin").Cells(VarNeueZeile, 3).NumberFormat = "@"
   Worksheets("tblTilgungEin").Cells(VarNeueZeile, 3).Value = Me.txtPLZ.Text
   Worksheets("tblTilgungEin").Cells(VarNeueZeile, 4).NumberFormat = "@"
   Worksheets("tblTilgungEin").Cells(VarNeueZeile, 4).Value = Me.txtOrt.Text
   With Worksheets("tblTilgungEin")
      .Range(.Cells(1, 1), .Cells(VarNeueZeile, 4)).Sort Key1:= _
      Worksheets("tblTilgungEin").Columns("A")
   End With
   varWelcherEintrag = VarNeueZeile
End Sub
```

```
Private Sub cmdAusgeben_Click()
   If Me.cboFirma.Text > "" And Me.txtAnschrift.Text > "" And Me.txtPLZ.Text > "" _
      And Me.txtOrt.Text > "" And Me.cboLaufzeit.Text > "" And Me.txtKredit.Text > "" Then
      Worksheets("tblTilgungAus").Activate
      Ausgabeloeschen
      Dim varZinssatz As Double
      Worksheets("tblTilgungAus").Cells(2, 1).NumberFormat = "@"
      Worksheets("tblTilgungAus").Cells(2, 1).Value = "Firma:"
      Worksheets("tblTilgungAus").Cells(2, 2).NumberFormat = "@"
      Worksheets("tblTilgungAus").Cells(2, 2).Value = Me.cboFirma.Text
      Worksheets("tblTilgungAus").Cells(3, 1).NumberFormat = "@"
      Worksheets("tblTilgungAus").Cells(3, 1).Value = "Anschrift:"
      Worksheets("tblTilgungAus").Cells(3, 2).NumberFormat = "@"
      Worksheets("tblTilgungAus").Cells(3, 2).Value = Me.txtAnschrift.Text
      Worksheets("tblTilgungAus").Cells(4, 1).NumberFormat = "@"
      Worksheets("tblTilgungAus").Cells(4, 1).Value = "PLZ + Ort:"
```

```
Worksheets("tblTilgungAus").Cells(4, 2).NumberFormat = "@"
Worksheets("tblTilgungAus").Cells(4, 2).Value = Me.txtPLZ.Text & " " & Me.txtOrt.Text
Worksheets("tblTilgungAus").Cells(5, 1).NumberFormat = "@"
Worksheets("tblTilgungAus").Cells(5, 1).Value = "Kredithöhe:"
Worksheets("tblTilgungAus").Cells(5, 2).NumberFormat = "#,##0.00 $"
Worksheets("tblTilgungAus").Cells(5, 2).Value = Me.txtKredit.Text
If Me.optANK.Value = True Then
   Worksheets("tblTilgungAus").Cells(6, 1).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(6, 1).Value = "Kreditart:"
   Worksheets("tblTilgungAus").Cells(6, 2).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(6, 2).Value = "Anleihenkredit"
   Worksheets("tblTilgungAus").Cells(7, 1).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(7, 1).Value = "Zinssatz:"
   Worksheets("tblTilgungAus").Cells(7, 2).NumberFormat = "0.000%"
   varZinssatz = 0.02
   Worksheets("tblTilgungAus").Cells(7, 2).Value = varZinssatz
End If
If Me.optHYK.Value = True Then
   Worksheets("tblTilgungAus").Cells(6, 1).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(6, 1).Value = "Kreditart:"
   Worksheets("tblTilgungAus").Cells(6, 2).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(6, 2).Value = "Hypothekenkredit"
   Worksheets("tblTilgungAus").Cells(7, 1).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(7, 1).Value = "Zinssatz:"
   Worksheets("tblTilgungAus").Cells(7, 2).NumberFormat = "0.000%"
   varZinssatz = 0.034
   Worksheets("tblTilgungAus").Cells(7, 2).Value = varZinssatz
End If
If Me.optPLK.Value = True Then
   Worksheets("tblTilgungAus").Cells(6, 1).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(6, 1).Value = "Kreditart:"
   Worksheets("tblTilgungAus").Cells(6, 2).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(6, 2).Value = "Policenkredit"
   Worksheets("tblTilgungAus").Cells(7, 1).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(7, 1).Value = "Zinssatz:"
   Worksheets("tblTilgungAus").Cells(7, 2).NumberFormat = "0.000%"
   varZinssatz = 0.049
   Worksheets("tblTilgungAus").Cells(7, 2).Value = varZinssatz
End If
If Me.optUNK.Value = True Then
   Worksheets("tblTilgungAus").Cells(6, 1).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(6, 1).Value = "Kreditart:"
   Worksheets("tblTilgungAus").Cells(6, 2).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(6, 2).Value = "Unternehmenskredit"
   Worksheets("tblTilgungAus").Cells(7, 1).NumberFormat = "@"
   Worksheets("tblTilgungAus").Cells(7, 1).Value = "Zinssatz:"
   Worksheets("tblTilgungAus").Cells(7, 2).NumberFormat = "0.000%"
   varZinssatz = 0.073
   Worksheets("tblTilgungAus").Cells(7, 2).Value = varZinssatz
End If
Worksheets("tblTilgungAus").Cells(8, 1).NumberFormat = "@"
Worksheets("tblTilgungAus").Cells(8, 1).Value = "Laufzeit im Jahr"
Worksheets("tblTilgungAus").Cells(8, 2).NumberFormat = "0 ""Jahr"""
Worksheets("tblTilgungAus").Cells(8, 2).Value = Me.cboLaufzeit.Text
Worksheets("tblTilgungAus").Cells(10, 1).NumberFormat = "@"
Worksheets("tblTilgungAus").Cells(10, 1).Value = "Jahr"
Worksheets("tblTilgungAus").Cells(10, 2).NumberFormat = "@"
Worksheets("tblTilgungAus").Cells(10, 2).Value = "Restschuld Kt"
Worksheets("tblTilgungAus").Cells(10, 3).NumberFormat = "@"
Worksheets("tblTilgungAus").Cells(10, 3).Value = "Zinszahlung Zt"
Worksheets("tblTilgungAus").Cells(10, 4).NumberFormat = "@"
Worksheets("tblTilgungAus").Cells(10, 4).Value = "Tilgung Tt"
Worksheets("tblTilgungAus").Cells(10, 5).NumberFormat = "@"
Worksheets("tblTilgungAus").Cells(10, 5).Value = "Annuität At"
Dim varRestkredit As Currency, varTilgung As Currency
varKredit = Me.txtKredit.Text
```

```
        varTilgung = 0
        varRestkredit = 0
        varTilgung = Me.txtKredit.Text / Me.cboLaufzeit.Text
        i = 1
        For i = 1 To Me.cboLaufzeit.Text
            If i = 1 Then
                varRestkredit = Me.txtKredit.Text
            Else
                varRestkredit = varRestkredit - varTilgung
            End If
            Worksheets("tblTilgungAus").Cells(10 + i, 1).Value = i
            Worksheets("tblTilgungAus").Cells(10 + i, 2).NumberFormat = "#,##0.00 $"
            Worksheets("tblTilgungAus").Cells(10 + i, 2).Value = varRestkredit
            Worksheets("tblTilgungAus").Cells(10 + i, 3).NumberFormat = "#,##0.00 $"
            Worksheets("tblTilgungAus").Cells(10 + i, 3).Value = varRestkredit * varZinssatz
            Worksheets("tblTilgungAus").Cells(10 + i, 4).NumberFormat = "#,##0.00 $"
            Worksheets("tblTilgungAus").Cells(10 + i, 4).Value = varTilgung
            Worksheets("tblTilgungAus").Cells(10 + i, 5).NumberFormat = "#,##0.00 $"
            Worksheets("tblTilgungAus").Cells(10 + i, 5).Value = varRestkredit * varZinssatz + varTilgung
        Next  i
    Else
        MsgBox "Sie müssen einen Eintrag wählen, oder ein Feld ist leer!"
    End If
End Sub
```

```
Sub Ausgabeloeschen()
    For i = 0 To 6
        Worksheets("tblTilgungAus").Cells(2 + i, 2).Value = Null
    Next i
    For i = 0 To 30
        Worksheets("tblTilgungAus").Cells(i + 10, 1).Value = Null
        Worksheets("tblTilgungAus").Cells(i + 10, 2).Value = Null
        Worksheets("tblTilgungAus").Cells(i + 10, 3).Value = Null
        Worksheets("tblTilgungAus").Cells(i + 10, 4).Value = Null
        Worksheets("tblTilgungAus").Cells(i + 10, 5).Value = Null
    Next i
End Sub
```

```
Private Sub cmdkennlinien_Click()
    Sheets("DTilgungAus").Select
    ActiveChart.Axes(xlValue).AxisTitle.Select
    Selection.Caption = "Wert in EURO"
    ActiveChart.Axes(xlCategory).AxisTitle.Select
    Selection.Caption = "Jahr"
    ActiveChart.ChartArea.Select
    ActiveChart.ChartTitle.Select
    If Me.optANK.Value = True Then
        ActiveChart.ChartTitle.Text = "Anleihenkredit"
    ElseIf Me.optHYK.Value = True Then
        ActiveChart.ChartTitle.Text = "Hypothekenkredit"
    ElseIf Me.optPLK.Value = True Then
        ActiveChart.ChartTitle.Text = "Policenkredit"
    ElseIf Me.optUNK.Value = True Then
        ActiveChart.ChartTitle.Text = "Unternehmenkredit"
    Else
        ActiveChart.ChartTitle.Text = "Keine Kredit gewählt"
    End If
End Sub
```

```
Private Sub cmdBeenden_Click()
    Me.Hide
End Sub
```

```
Private Sub cmdInfo_Click()
    Worksheets("tblTilgungInfo").Activate
End Sub
```

A3.7) Wie schon im Abschnitt A3.2.6 erläutert wurde, muss das Programm als Ergebnis Abb. A3.7a für den Fall „Ausgeben“ liefern.

	A	B	C	D	E
1	Tilgungsrechnung für die Kapitalanlagen				
2	**Firma:**	H&M AG			
3	**Anschrift:**	Postfach 2318			
4	**PLZ + Ort:**	D-36243 Niederaula			
5	**Kredithöhe:**	250.000,00 €			
6	**Kreditart:**	Unternehmenskredit			
7	**Zinssatz:**	7,300%			
8	**Laufzeit im Jahr**	20 Jahr			
9					
10	**Jahr**	**Restschuld Kt**	**Zinszahlung Zt**	**Tilgung Tt**	**Annuität At**
11	1	250.000,00 €	18.250,00 €	12.500,00 €	30.750,00 €
12	2	237.500,00 €	17.337,50 €	12.500,00 €	29.837,50 €
13	3	225.000,00 €	16.425,00 €	12.500,00 €	28.925,00 €
14	4	212.500,00 €	15.512,50 €	12.500,00 €	28.012,50 €
15	5				
16	6				
17	7				
18	8				
19	9				
20	10				
21	11				
22	12				
23	13				
24	14				
25	15				
26	16				
27	17				
28	18				
29	19				
30	20				
31					
32					
33					
34					
35					
36					

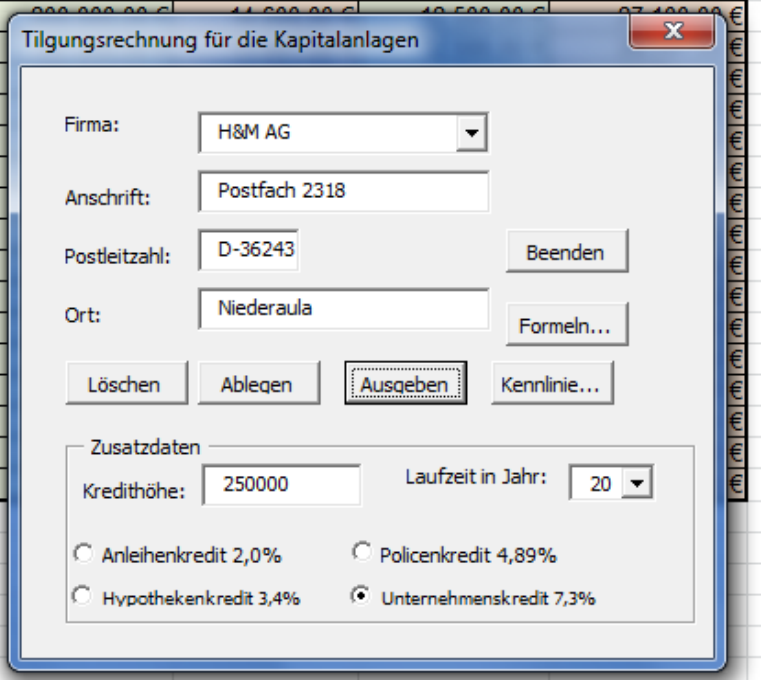


Abb. A3.7a Das Layout beim Ausgeben

A3.8) **LITERATUR**

- Investition und Finanzierung von Ulrich Ermschel et al., ISBN-978-3-7908-2745-3, Springer-Verlag Berlin Heidelberg 2011(https://kataloge.uni-hamburg.de/DB=2/SET=5/TTL=1/SHW?FRST=3)
- Automatisierung und Programmierung mit Excel 2010 (www.rrzn.uni-hannover.de/buecher.html)
- VBA – Programmierung mit Excel von Johannes Gogolok (VBA mit Excel von Uni-Hagen.pdf)

A4) Aufgabe: Eine Datenbank für Materialkalkulation / Stückliste

Kurzerklärung
Stückliste beschreibt die mengenmäßige Zusammensetzung eines Erzeugnisses aus seinen Einzelteilen. Die Stückliste gibt dabei an, wie viele Mengeneinheiten eines bestimmten Teils oder einer bestimmten Baugruppe auf untergeordneter Erzeugnisstrukturebene benötigt werden, um eine Einheit des Erzeugnisses auf übergeordneter Erzeugnisstrukturebene herzustellen. wichtige Angaben (bez. der untergeordneten Teile) sind dabei Teilenummer, evtl. Variantennummer, Teilebezeichnung, Mengenkoeffizient, evtl. Verweis auf zugehörige Konstruktionszeichnung u.a. Stückliste wird an vielen Stellen eines Produktionsbetriebs bei der deterministischen Bedarfsplanung, Konstruktion, Vor- und Nachkalkulation verwendet.

Beschreibung

Erstellen Sie eine Datenbank, mit der eine Liste der Materialien und deren kalkulaktionsdaten verwaltet und ermittelt werden. In dieser Aufgabestellung geht es u.a darum, mit einer Userform (Abb. A4.2a und Abb. A4.2b) die Kundendaten (Tab. A4.1a) und die Materialdaten (Tab. A4.1b) zu verwalten, sie mit den Kalkulationsformeln (Abb. A4.5a) in eine Ausgabetabelle (Tab. A4.7a) zu übertragen. Ein leerer Entwurf der Ausgabetabelle (Tab. A4.3a) soll mithilfe eines Macros aufgezeichnet werden.

Wichtige Funktionen
Aufbau einer Exceldatei zu einer automatisierten Materialstückliste mit folgenden Funktionen:

- Umgang mit zwei Stammdatentabellen und einer Userform
- Programmierung und Umgang mit den zwei bzw. mehreren Datensätzen
- Konzeption und Realisation der Algorithmen zu der Aufgabestellung
- Umgang mit den Macroaufzeichnungen und benutzerdefinierten Prozeduren
- Berücksichtigung der Qualitätsfaktoren (Korrektheit, Robustheit, benutzerfreunlichkeit uws)

Vorgehensweise

Erstellen Sie eine Exceldatei mit dem Namen axxxxxxx.xLsm (xxxxxxx steht für Ihre Matrikelnummer). In dieser Aufgabe werden drei Tabellen (`tblAKKEIN` für die Kundendaten (Tab. A4.1a) und `tblMKEIN` für die Materialdaten(Tab. A4.1b) als Stammdatentabellen und `tblMKAUS` als Ausgabetabelle (Tab. A4.7a)) und eine Userform mit dem Namen `frmMaterial` benötigt. Die Userform (Abb. A4.2a und Abb. A4.2b) muss aus zwei Seiten bestehen, damit jede Seite für die Verwaltung der Datensätze in der entsprechenden Stammdatentabelle dargestellt wird.

A4.1.1) Das Tabellenblatt (Tab. A4.1a) kann `tblAKKEIN` heißen und beinhaltet Datensätze für die `Auftragsnummer` (Spalte A), `Firma` (Spalte B), `Straße` (Spalte C), `Postleitzahl` (Spalte D) und den `Ort` (Spalte E).

A4.1.2) Das Tabellenblatt (Tab. A4.1b) kann `tblMKEIN` heißen und beinhaltet Datensätze für die `Bausteinnummer` (Spalte A), `Bausteinbezeichnung` (Spalte B), den `Preis` (Spalte C), die `Grundmenge` (Spalte D), den `Faktor` (Spalte E) und die `Istmenge` (Spalte F).

	A	B	C	D	E
1	3452	Atelier Freier Fall	Methfesselstr. 16a	20257	Hamburg
2	3458	Beutin	Jungfernstieg 11/12	20354	Hamburg
3	3481	Beutin	Breite Straße 91	23552	Lübeck
4	3492	Come Clother	Brauerknechtgraben 53	20459	Hamburg
5	3495	Cashmere House	Jägerstraße 61	10117	Berlin
6					
7	**Auftrags-nummer**	**Firma**	**Straße**	**PLZ**	**Ort**

Tab. A4.1a `tblAKKEIN`

	A	B	C	D	E	F
1	301	DPT mit Aufschlag	21,30 €	11	1,0150	12
2	311	FlügelT mit Aufschlag	22,40 €	6	1,0210	5
3	312	FlügelT ohne Aufschlag	21,40 €	8	1,0000	4
4	901	Klassisch	5,50 €	5	1,0000	5
5	902	Business1	6,30 €	4	1,0010	9
6	904	City1	9,20 €	9	1,0500	9
7	911	Vpasse	9,55 €	7	1,0220	6
8						
9	**Baustein-nummer**	**Bausteinbezeichnung**	**Preis**	**Grund-Menge**	**Faktor**	**Ist-Menge**

Tab. A4.1b `tblMKEIN`

A4.2) Das Userformblatt (Abb. A4.2a und Abb. A4.2b) kann `frmMaterial` heißen und besteht aus einem Steuerelement „ Multiseiten“ mit zwei Seiten. Die erste Seite für die Kundendaten und die zweite Seite für die Materialdaten sollen vorgesehen werden. Die notwendigen Steuerelemente für jede Seite werden wie folgt zusammengefasst:

Steuerelemente für die Kundendaten

Pos.	Bezeichnung (Caption)	Steuerelement	Name
1	Auftragsnummer:	Beschriftungsfeld	Nicht ändern
2	Keine	Kombinationsfeld	cboAtnr
3	Firma:	Beschriftungsfeld	Nicht ändern
4	Keine	Textfeld	txtFirma
5	Straße + Hausnummer:	Beschriftungsfeld	Nicht ändern
6	Keine	Textfeld	txtStrasse
7	Postleitzahl:	Beschriftungsfeld	Nicht ändern
8	keine	Textfeld	txtPLZ
9	Ort:	Beschriftungsfeld	Nicht ändern
10	keine	Textfeld	txtOrt
11	Ablegen	Befehlsschaltfläche	cmdKAblegen

12	Löschen	Befehlsschaltfläche	cmdKLoeschen
13	Beenden	Befehlsschaltfläche	cmdKBeenden

Steuerelemente für die Materialdaten

Pos.	Bezeichnung (Caption)	Steuerelement	Name
1	Bausteinnummer:	Beschriftungsfeld	Nicht ändern
2	Keine	Kombinationsfeld	cboBstn
3	Bausteinbezeichnung:	Beschriftungsfeld	Nicht ändern
4	Keine	Textfeld	txtBsbez
5	Datum:	Beschriftungsfeld	Nicht ändern
6	Keine	Textfeld	txtDatum
7	Preis:	Beschriftungsfeld	Nicht ändern
8	keine	Textfeld	txtPreis
9	Grundmenge:	Beschriftungsfeld	Nicht ändern
10	keine	Textfeld	txtGrundmenge
11	Faktor:	Beschriftungsfeld	Nicht ändern
12	keine	Textfeld	txtFaktor
13	Istmenge:	Beschriftungsfeld	Nicht ändern
14	keine	Textfeld	txtIstmenge
15	Ablegen	Befehlsschaltfläche	cmdMAblegen
16	Löschen	Befehlsschaltfläche	cmdMLoeschen
17	Ausgeben	Befehlsschaltfläche	cmdMAusgeben
18	Forrmeln…	Befehlsschaltfläche	cmdMInfo
19	Beenden	Befehlsschaltfläche	cmdMBeenden

Erläuterung zu dem Programmablauf für die Kundendaten

A4.2.1.1) Beim Starten des Userformblatts (Abb. A4.2a) müssen alle Einträge der ersten Spalte von der Tabelle `tblAKKEIN` (Tab. A4.1a) in das Kombinationsfeld `cboAtnr` eingefügt werden.

A4.2.1.2) Wenn ein Eintrag für die Auftragsnummer gewählt wird, muss der entsprechende Datensatz aus der Tabelle `tblAKKEIN` in den Textfeldern erscheinen.

A4.2.1.3) Wenn der Benutzer einen neuen Eintrag in das Userformblatt (Abb. A4.2a) einträgt und auf „Ablegen“ klickt, muss der neue Datensatz in die Tabelle `tblAKKEIN` hinzugefügt werden und anschließend sortiert werden.

Erläuterung zu dem Programmablauf für die Materialdaten

A4.2.2.1) Beim Starten des Userformblatts (Abb. A4.2b) müssen alle Einträge der ersten Spalte von der Tabelle `tblMKEIN` (Tab. A4.1b) in das Kombinationsfeld `cboBstn` eingefügt werden.

A4.2.2.2) Wenn ein Eintrag für die Bausteinummer gewählt wird, muss der entsprechende Datensatz aus der Tabelle `tblMKEIN` in den Textfeldern erscheinen.

A4.2.2.3) Wenn der Benutzer einen neuen Eintrag in das Userformblatt (Abb. A4.2b) einträgt und auf „Ablegen“ klickt, muss der neue Datensatz in die Tabelle `tblMKEIN` hinzugefügt werden und anschließend sortiert werden.

A4.2.1.4) Wenn der Benutzer einen Eintrag wählt, der schon in der Tabelle `tblAKKEIN` existiert, muss der Datensatz beim Klicken auf „Ablegen“ ersetzt werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz ersetzt werden soll.)
A4.2.1.5) Wenn der Benutzer einen Eintrag wählt und auf die Befehlsschaltfläche „Löschen“ klickt, muss der Datensatz aus der Tabelle `tblAKKEIN` gelöscht werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz gelöscht werden soll.)
A4.2.1.7) Die Befehlsschaltfläche „Beenden“ blendet die Userform aus.

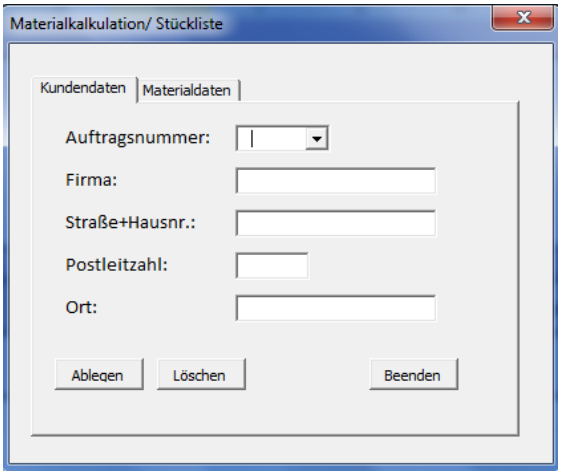


Abb. A4.2a `frmMaterial Kundendaten`

Abb. A4.2b `frmMaterial Materialdaten`

A4.2.2.4) Wenn der Benutzer einen Eintrag wählt, der schon in der Tabelle `tblMKEIN` existiert, muss der Datensatz beim Klicken auf „Ablegen“ ersetzt werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz ersetzt werden soll.)
A4.2.2.5) Wenn der Benutzer einen Eintrag wählt und auf die Befehlsschaltfläche „Löschen“ klickt, muss der Datensatz aus der Tabelle `tblMKEIN` gelöscht werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz gelöscht werden soll.)
A4.2.2.6) Wenn der Benutzer auf die Befehlsschaltfläche „Ausgeben“ klickt, müssen alle in der Userform `frmMaterial` ausgewälten Daten mit den berechneten Werten für die Gesamtmenge, Plansumme, Istsumme, Abweichung, Gesamtplan, GesamtIst und die GesamtAbweichung (Abb. A4.5a) in die Ausgabetabelle(Tab. A4.3a) `tblMKAUS` übertragen werden. Hierbei sollen insbesonders die Softwarequalitäten berücksichtigt werden. Z.B. sollen Sicherheitsmeldungen ausgegeben, wenn der Benutzer keinen Eintrag gewählt hat, wenn die Inhalte der Textfelder ungültige Datentypen beinhalten.
A4.2.2.7) Die Befehlsschaltfläche „Beenden“ blendet die Userform aus.

A4.3) Erstellen Sie eine Tabelle (Tab. A4.3a) mit dem Namen `tblMKAus`. Zeichen Sie dann ein Macro mit dem Namen `TmsAufbauen` für dieses Tabellenblatt mit folgenden Merkmalen:

- Die Spaltenbreiten (A=180 Pixel, B bis I=75 Pixel und J=96 Pixel) und die Zeilenhöhen (Zeile1=30 Pixel, Zeile2-8=24 Pixel, Zeile9 = 40 Pixel und Zeile10,11= 24 Pixel).
- Die Hintergrundfarben von den Zellen A1-J1 und A9-J9 sind Olivgrün, Akzent 3, dunkler 25%, von den Zellen A2-J7 und A10-J11 sind Olivgrün, Akzent 3, heller 80% und von den Zellen A8-J8 sind Olivgrün, Akzent 3, heller 60%.
- Der äußere Rahmen und die vertikal und horizontal inneren Rahmen haben die Breite 1 ½ Pt und die Farbe schwarz, Text 1, heller 5%.
- Alle Zellen haben eine Schriftart Calibri, einen Schriftschnitt fett und einen Schriftgrad 10. Die Zellen A1 und A9-J9 haben eine Schriftfabe weiß, Hintergrund 1, dunkler 5%, Die Zellen B3-B7, H3-H7 und B8 haben eine Schriftfabe schwarz, Text 1, heller 5% und die Zellen C3-C7, I3-I8 und A10-J11 haben eine Schriftfabe rot, Akzent 2, dunkler 50%.
- Es wird dauf hingewiesen, dass die Zellenformatierungen für die Ausgabedaten in Codes vorgenommen werden müssen.

	A	B	C	D	E	F	G	H	I	J
1	**Materialkalkulation/Stückliste**									
2										
3							**Firma:**			**Materialeinsatz**
4							**Anschrift:**			**GesamtPlan:**
5										**GesamtIst:**
6										**GesamtAbweichung:**
7							**Auftragsnummer:**			**Datum:**
8	**Eingabedaten**									
9	**Bezeichnung**	**Baustein-nummer**	**Preis**	**Grund-menge**	**Faktor**	**Ist-menge**	**Gesamt-menge**	**Plan-summe**	**Ist-summe**	**Abweichung in %**
10										
11										

tblAKKEIN | tblAKINFO | tblAKAEIN | tblAKAUS | tblMKINFO | tblMKEIN | **tblMKAUS**

Tab. A4.3a `tblMKAus`

Abb. A4.5a

Formeln und Erklärung:

Formelsammlung		
GesamtMenge	=	GrundMenge * Faktor
PlanSumme	=	GesamtMenge * Preis
IstSumme	=	IstMenge * Preis
Abweichung in %	=	(PlanSumme - IstSumme) / PlanSumme
GesamtPlan	=	Summe der PlanSumme
GesamtIst	=	Summe der IstSumme
GesamtAbweichung	=	GesamtPlan - GesamtIst

Programmablaufplan für die Kundendaten

Der Programmablaufplan (Abb. A4.6a) bezieht sich auf die Abschnitten A4.2.1.1 – A4.2.1.7 Entwickeln und realisieren Sie bitte einen Programmablaufplan und ein Programmcode, in dem kurze Codes und vereinfachte Aktionen für den Benutzer ermöglicht werden.

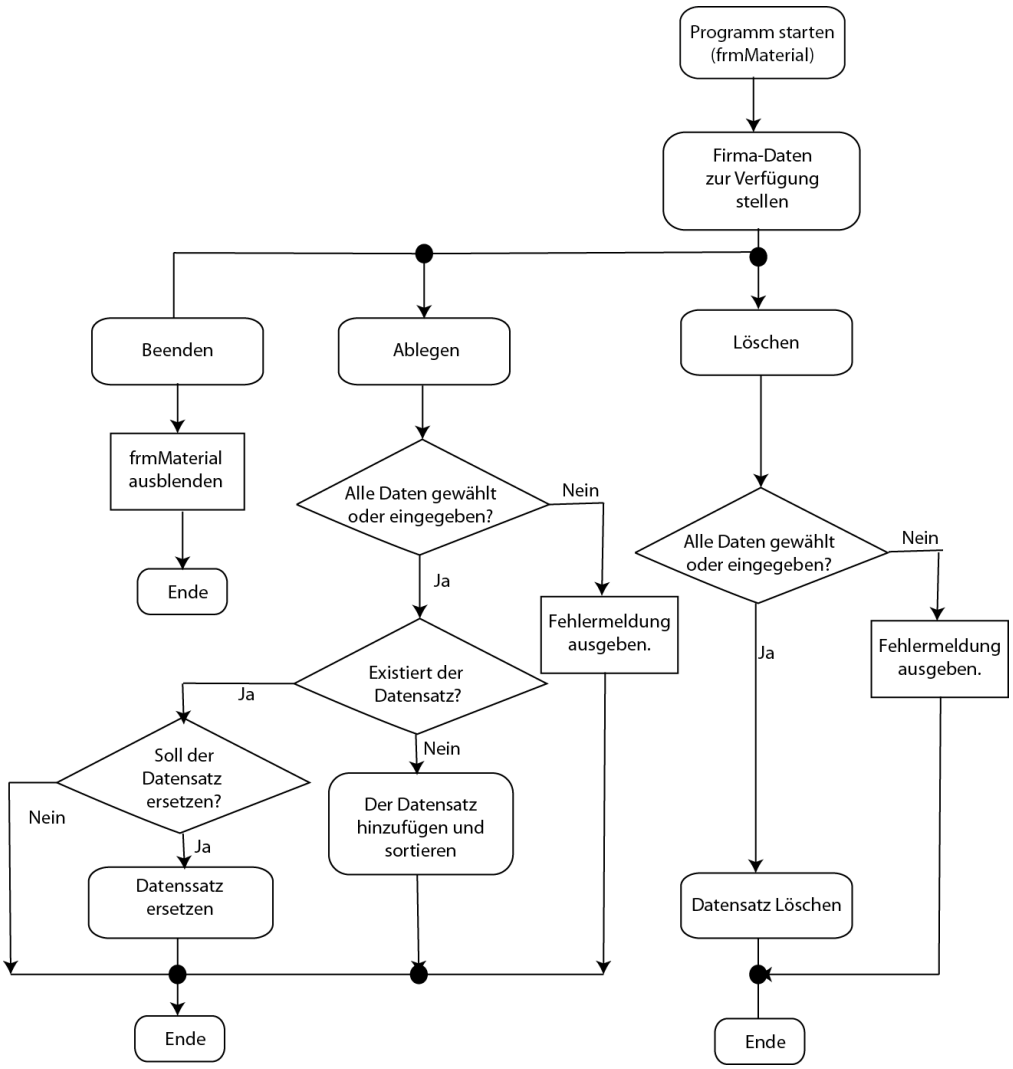


Abb. A4.6a Programmablaufplan zu der kundendaten

Programmablaufplan für die Kundendaten

Der Programmablaufplan (Abb. A4.6b) bezieht sich auf die Abschnitten A4.2.2.1 – A4.2.2.7 Entwickeln und realisieren Sie bitte einen Programmablaufplan und ein Programmcode, in dem kurze Codes und vereinfachte Aktionen für den Benutzer ermöglicht werden.

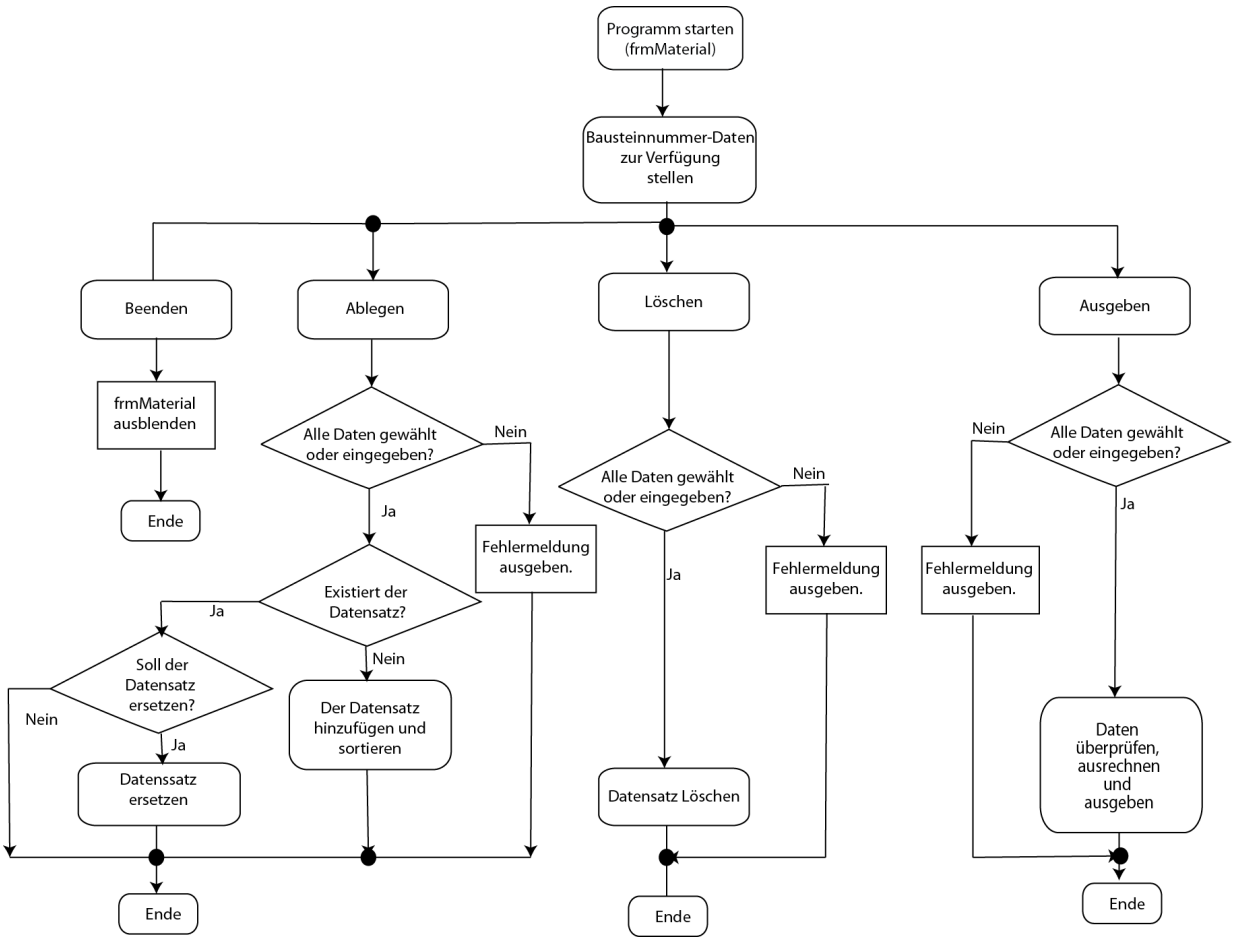


Abb. A4.6b Programmablaufplan zu der Materialdaten

A4.6.1) Das Macro `TmsAufbauen` kann etwa so aussehen:

```
Sub TmsAufbauen()
   Columns("A:A").ColumnWidth = 25
   Columns("B:B").ColumnWidth = 10
   Columns("C:C").ColumnWidth = 10
   Columns("D:D").ColumnWidth = 10
   Columns("E:E").ColumnWidth = 10
   Columns("F:F").ColumnWidth = 10
   Columns("G:G").ColumnWidth = 10
   Columns("H:H").ColumnWidth = 10
   Columns("I:I").ColumnWidth = 10
   Columns("J:J").ColumnWidth = 13
   Rows("1:1").RowHeight = 21.75
   Rows("2:2").Select
   Selection.RowHeight = 18
   Rows("3:3").RowHeight = 18
   Rows("4:4").RowHeight = 18
   Rows("5:5").RowHeight = 18
   Rows("6:6").RowHeight = 18
   Rows("7:7").RowHeight = 18
   Rows("8:8").RowHeight = 18
   Rows("9:9").RowHeight = 18
```

```
    Rows("10:10").RowHeight = 18
    Rows("11:11").RowHeight = 30
    Rows("12:12").RowHeight = 18
    Rows("13:13").RowHeight = 18
    Range("A1:J1").Select
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlMedium
    End With
    .....
    ...
```

Aus den Platzgründen wurden die Codes gekürtzt.

```
    ....
    .....
    With Selection.Font
        .Name = "Calibri"
        .Size = 11
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ThemeColor = xlThemeColorLight1
        .TintAndShade = 0
        .ThemeFont = xlThemeFontMinor
    End With
    Selection.Font.Bold = True
    With Selection.Font
        .ThemeColor = xlThemeColorAccent2
        .TintAndShade = -0.249977111117893
    End With
    Range("H13").Select
End Sub
```

A4.6.2)Folgende VBA-Codes dienen dazu, um die in den Abschnitten A4.2.1.1-A4.2.1.7 und A4.2.2.1-A4.2.2.7 Programmablaufpläne zu implementieren:

```
Dim varKWelcherEintrag As Integer  'Eintrag für Auftragsnummer
Dim varNeuerEintrag As Integer
Dim varMWelcherEintrag As Integer  'Eintrag für varBausteinnummer
Dim varGesamtPlan As Currency
Dim varGesamtIst As Currency
```

```
Private Sub UserForm_Initialize()
    varNeuerEintrag = 0
    varGesamtPlan = 0
    varGesamtIst = 0
    KMaskeloeschen
    KEinfuegen
    MMaskeloeschen
    MEinfuegen
    Me.txtDatum.Text = Date
    MultiPage1.Pages(0).Enabled = True
End Sub
```

```
Sub KEinfuegen()
    Dim varAuftragsnummer As Integer
    varAuftragsnummer = Worksheets("tblAKKEIN").Cells(1, 1).CurrentRegion.Rows.Count
    For i = 1 To varAuftragsnummer
        Me.cboAtnr.AddItem Worksheets("tblAKKEIN").Cells(i, 1).Text
    Next i
End Sub
```

```
Sub MEinfuegen()
    Dim varBausteinnummer As Integer
    varBausteinnummer = Worksheets("tblMKEIN").Cells(1, 1).CurrentRegion.Rows.Count
    For i = 1 To varBausteinnummer
        Me.cboBstn.AddItem Worksheets("tblMKEIN").Cells(i, 1).Text
    Next i
End Sub
Sub KMaskeloeschen()
    Me.cboAtnr.Clear
    Me.cboAtnr.Text = ""
    Me.txtFirma.Text = ""
    Me.txtStrasse.Text = ""
    Me.txtPlz.Text = ""
    Me.txtOrt.Text = ""
End Sub
Sub MMaskeloeschen()
    Me.cboBstn.Clear
    Me.cboBstn.Text = ""
    Me.txtBsbez.Text = ""
    Me.txtPreis.Text = ""
    Me.txtGrundmenge.Text = ""
    Me.txtFaktor.Text = ""
    Me.txtIstmenge.Text = ""
End Sub
Private Sub cmdKBeenden_Click()
    varNeuerEintrag = 0
    ActiveWindow.WindowState = xlNormal
    Me.Hide
End Sub
Private Sub cmdMBeenden_Click()
    varNeuerEintrag = 0
    ActiveWindow.WindowState = xlNormal
    Me.Hide
End Sub
Private Sub cboAtnr_Click()
    varKWelcherEintrag = Me.cboAtnr.ListIndex + 1
    Me.txtFirma.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 2).Value
    Me.txtStrasse.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 3).Value
    Me.txtPlz.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 4).Value
    Me.txtOrt.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 5).Value
End Sub
Private Sub cboAtnr_Enter()
    Me.txtFirma.Text = ""
    Me.txtStrasse.Text = ""
    Me.txtPlz.Text = ""
    Me.txtOrt.Text = ""
End Sub
Private Sub cboAtnr_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Me.txtFirma.Text = ""
    Me.txtStrasse.Text = ""
    Me.txtPlz.Text = ""
    Me.txtOrt.Text = ""
End Sub
Private Sub cboBstn_Click()
    varMWelcherEintrag = Me.cboBstn.ListIndex + 1
    Me.txtBsbez.Text = Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 2).Value
    Me.txtPreis.Text = Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 3).Value
    Me.txtGrundmenge.Text = Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 4).Value
    Me.txtFaktor.Text = Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 5).Value
    Me.txtIstmenge.Text = Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 6).Value
End Sub
Private Sub cboBstn_Enter()
    Me.txtBsbez.Text = ""
    Me.txtPreis.Text = ""
    Me.txtGrundmenge.Text = ""
```

```
    Me.txtFaktor.Text = ""
    Me.txtIstmenge.Text = ""
End Sub
Private Sub cboBstn_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Me.txtBsbez.Text = ""
    Me.txtPreis.Text = ""
    Me.txtGrundmenge.Text = ""
    Me.txtFaktor.Text = ""
    Me.txtIstmenge.Text = ""
End Sub
Private Sub cmdKLoeschen_Click()
    If Me.cboAtnr.Text = "" Then
        Beep
        MsgBox "Sie müssen einen Eintrag wählen."
    Else
        Mldg = "Wollen Sie wirklich diesen Datensatz loeschen?"
        Stil = vbYesNo + vbCritical + vbDefaultButton2
        Title = "Datensatz loeschen"
        Ergebnis = MsgBox(Mldg, Stil, Title)
        If Ergebnis = vbYes Then
            Worksheets("tblAKKEIN").Rows(varKWelcherEintrag).Delete Shift:=xlUp
            Worksheets("tblAKKEIN").Activate
            KMaskeloeschen
            KEinfuegen
        End If
    End If
End Sub
Private Sub cmdMloeschen_Click()
    If Me.cboBstn.Text = "" Then
        Beep
        MsgBox "Sie müssen einen Eintrag wählen."
    Else
        Mldg = "Wollen Sie wirklich diesen Datensatz löschen?"
        Stil = vbYesNo + vbCritical + vbDefaultButton2
        Title = "Datensatz loeschen"
        Ergebnis = MsgBox(Mldg, Stil, Title)
        If Ergebnis = vbYes Then
            Worksheets("tblMKEIN").Rows(varMWelcherEintrag).Delete Shift:=xlUp
            Worksheets("tblMKEIN").Activate
            MMaskeloeschen
            MEinfuegen
        End If
    End If
End Sub
Private Sub cmdkAblegen_Click()
    If Me.cboAtnr.Text = "" Then
        MsgBox "Sie müssen einen Eintrag wählen."
    Else
        If Me.cboAtnr.Text > "" And varKWelcherEintrag > 0 Then
            If Me.cboAtnr.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 1).Value Then
                Mldg = "Wollen Sie diesen Datensatz ersetzen?"
                Stil = vbYesNo + vbCritical + vbDefaultButton2
                Title = "Kundendaten"
                Ergebnis = MsgBox(Mldg, Stil, Title)
                If Ergebnis = vbYes Then
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 1).Value = Me.cboAtnr.Text
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 2).Value = Me.txtFirma.Text
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 3).Value = Me.txtStrasse.Text
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 4).Value = Me.txtPlz.Text
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 5).Value = Me.txtOrt.Text
                    Worksheets("tblAKKEIN").Activate
                    KMaskeloeschen
                    KEinfuegen
                    Exit Sub
                End If
            Else
```

```
            Worksheets("tblAKKEIN").Activate
            KNeueintragen
            KMaskeloeschen
            KEinfuegen
         End If
      ElseIf Me.cboAtnr.Text > "" And varKWelcherEintrag = 0 Then
         Worksheets("tblAKKEIN").Activate
         KNeueintragen
         KMaskeloeschen
         KEinfuegen
      Else
         MsgBox "Der Fall komm gar nicht vor!"
         Exit Sub
      End If
   End If
End Sub
```

```
Private Sub cmdMAblegen_Click()
   If Me.cboBstn.Text = "" Then
      MsgBox "Sie müssen einen Eintrag wählen."
   Else
      If Me.cboBstn.Text > "" And varMWelcherEintrag > 0 Then
         If Me.cboBstn.Text = Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 1).Value Then
            Mldg = "Wollen Sie diesen Datensatz ersetzen?"
            Stil = vbYesNo + vbCritical + vbDefaultButton2
            Title = "Materialstückliste"
            Ergebnis = MsgBox(Mldg, Stil, Title)
            If Ergebnis = vbYes Then
               Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 1).Value = Me.cboBstn.Text
               Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 2).Value = Me.txtBsbez.Text
               Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 3).Value = Me.txtPreis.Text
               Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 4).Value = Me.txtGrundmenge.Text
               Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 5).Value = Me.txtFaktor.Text
               Worksheets("tblMKEIN").Cells(varMWelcherEintrag, 6).Value = Me.txtIstmenge.Text
               Worksheets("tblMKEIN").Activate
               MMaskeloeschen
               MEinfuegen
               Exit Sub
            End If
         Else
            Worksheets("tblMKEIN").Activate
            MNeueintragen
            MMaskeloeschen
            MEinfuegen
         End If
      ElseIf Me.cboBstn.Text > "" And varMWelcherEintrag = 0 Then
         Worksheets("tblMKEIN").Activate
         MNeueintragen
         MMaskeloeschen
         MEinfuegen
      Else
         MsgBox "Der Fall kommt gar nicht vor!"
         Exit Sub
      End If
   End If
End Sub
```

```
Sub KNeueintragen()
   Dim varNeueZeile As Integer
   varNeueZeile = Worksheets("tblAKKEIN").Cells(1, 1).CurrentRegion.Rows.Count + 1
   Worksheets("tblAKKEIN").Cells(varNeueZeile, 1).NumberFormat = "@"
   Worksheets("tblAKKEIN").Cells(varNeueZeile, 1).Value = Me.cboAtnr.Text
   Worksheets("tblAKKEIN").Cells(varNeueZeile, 2).NumberFormat = "@"
   Worksheets("tblAKKEIN").Cells(varNeueZeile, 2).Value = Me.txtFirma.Text
   Worksheets("tblAKKEIN").Cells(varNeueZeile, 3).NumberFormat = "@"
   Worksheets("tblAKKEIN").Cells(varNeueZeile, 3).Value = Me.txtStrasse.Text
   Worksheets("tblAKKEIN").Cells(varNeueZeile, 4).NumberFormat = "@"
   Worksheets("tblAKKEIN").Cells(varNeueZeile, 4).Value = Me.txtPlz.Text
```

```
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 4).NumberFormat = "@"
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 5).Value = Me.txtOrt.Text
    With Worksheets("tblAKKEIN")
       .Range(.Cells(1, 1), .Cells(varNeueZeile, 5)).Sort Key1:= _
          Worksheets("tblAKKEIN").Columns("A"), Header:=xlGuess
    End With
End Sub
```

```
Sub MNeueintragen()
    Dim varNeueZeile As Integer
    varNeueZeile = Worksheets("tblMKEIN").Cells(1, 1).CurrentRegion.Rows.Count + 1
    Worksheets("tblMKEIN").Cells(varNeueZeile, 1).Value.NumberFormat = "@"
    Worksheets("tblMKEIN").Cells(varNeueZeile, 1).Value = Me.cboBstn.Text
    Worksheets("tblMKEIN").Cells(varNeueZeile, 2).Value.NumberFormat = "@"
    Worksheets("tblMKEIN").Cells(varNeueZeile, 2).Value = Me.txtBsbez.Text
    Worksheets("tblMKEIN").Cells(varNeueZeile, 3).Value.NumberFormat = "0.00""€"""
    Worksheets("tblMKEIN").Cells(varNeueZeile, 3).Value = Me.txtPreis.Text
    Worksheets("tblMKEIN").Cells(varNeueZeile, 4).Value.NumberFormat = "0"
    Worksheets("tblMKEIN").Cells(varNeueZeile, 4).Value = Me.txtGrundmenge.Text
    Worksheets("tblMKEIN").Cells(varNeueZeile, 5).Value.NumberFormat = "0.0000"
    Worksheets("tblMKEIN").Cells(varNeueZeile, 5).Value = Me.txtFaktor.Text
    Worksheets("tblMKEIN").Cells(varNeueZeile, 6).Value.NumberFormat = "0"
    Worksheets("tblMKEIN").Cells(varNeueZeile, 6).Value = Me.txtIstmenge.Text
    With Worksheets("tblMKEIN")
       .Range(.Cells(1, 1), .Cells(varNeueZeile, 6)).Sort Key1:= _
          Worksheets("tblMKEIN").Columns("A"), Header:=xlGuess
    End With
End Sub
```

```
Private Sub cmdMAusgeben_Click()
  Rem Funktion Feldpruefen wird aufgerufen.
    If Felderpruefen Then
       Worksheets("tblMKAUS").Activate
       If varNeuerEintrag = 0 Then
          Mldg = "Ausgabetabelle  mit Ja löschen mit Nein leeren!"
          Stil = vbYesNo + vbCritical + vbDefaultButton2
          Title = "Stückliste löschen"
          Ergebnis = MsgBox(Mldg, Stil, Title)
          If Ergebnis = vbYes Then
            Rem Prozedure Taufbauen  Kopfteilausgeben ausführen.
             Taufbauen
             Kopfteilausgeben
          Else
            Rem Prozedure Tabelleleeren und Kopfteilausgeben ausführen.
             Tabelleleeren
             Kopfteilausgeben

          End If
          Rem die Seite "Kundendaten“  ausblenden
          MultiPage1.Pages(0).Enabled = False
       End If
       Rem Funktion Datenausgeben ausführen.
       Datenausgeben varNeuerEintrag
       varNeuerEintrag = varNeuerEintrag + 1
    Else
       Exit Sub
    End If
End Sub
```

```
Sub Taufbauen()
    If ActiveWorkbook.Worksheets("tblMKAUS").Activate Then
       Worksheets("tblMKAUS").Delete
       ActiveWorkbook.Worksheets.Add After:=Worksheets(Worksheets.Count)
       Worksheets(Worksheets.Count).Name = "tblMKAUS"
        Rem Macro TmsAufbauen ausführen.
       TmsAufbauen
    Else
       MsgBox "Tabelle existiert nicht!"
    End If
```

```
End Sub
Function Feldpruefen(varFeld As Object) As Variant
   If varFeld.Text > "" Then
      Feldpruefen = varFeld.Text
   Else
      MsgBox "Das Steuerelement " & varFeld.Name & " ist leer."
      varFeld.BackColor = RGB(255, 0, 0)
      Exit Function
   End If
End Function
Function Felderpruefen() As Boolean
   Dim varFeldMatrix(1 To 10) As Variant
   Dim varTypMatrix(1 To 10) As Variant
   Dim i As Integer
   i = 0
   For i = 1 To 10
      varFeldMatrix(i) = 0
      varTypMatrix(i) = 0
   Next i
   varFeldMatrix(1) = Feldpruefen(Me.cboBstn)
   varFeldMatrix(2) = Feldpruefen(Me.txtBsbez)
   varFeldMatrix(3) = Feldpruefen(Me.txtPreis)
   varFeldMatrix(4) = Feldpruefen(Me.txtGrundmenge)
   varFeldMatrix(5) = Feldpruefen(Me.txtFaktor)
   varFeldMatrix(6) = Feldpruefen(Me.txtIstmenge)
   varFeldMatrix(7) = Feldpruefen(Me.txtFirma)
   varFeldMatrix(8) = Feldpruefen(Me.txtStrasse)
   varFeldMatrix(9) = Feldpruefen(Me.txtPlz)
   varFeldMatrix(10) = Feldpruefen(Me.txtOrt)

   varTypMatrix(1) = Feldpruefen(Me.cboBstn)
   varTypMatrix(2) = Feldpruefen(Me.txtBsbez)
   varTypMatrix(3) = Feldpruefen(Me.txtPreis)
   varTypMatrix(4) = Feldpruefen(Me.txtGrundmenge)
   varTypMatrix(5) = Feldpruefen(Me.txtFaktor)
   varTypMatrix(6) = Feldpruefen(Me.txtIstmenge)
   varTypMatrix(7) = Feldpruefen(Me.txtFirma)
   varTypMatrix(8) = Feldpruefen(Me.txtStrasse)
   varTypMatrix(9) = Feldpruefen(Me.txtPlz)
   varTypMatrix(10) = Feldpruefen(Me.txtOrt)

   If varTypMatrix(1) = CStr(varFeldMatrix(1)) Then
      Felderpruefen = True
   Else
      MsgBox "Die Bausteinnummer muss eine Zeichenkette(String) sein!"
      Felderpruefen = False
      Exit Function
   End If
   If varTypMatrix(2) = CStr(varFeldMatrix(2)) And IsNumeric(varTypMatrix(2)) = False Then
      Felderpruefen = True
   Else
      MsgBox "Die Bausteinbezeichnung muss eine Zeichenkette(String) sein!"
      Felderpruefen = False
      Exit Function
   End If
   If varTypMatrix(3) = CCur(varFeldMatrix(3)) Then
      Felderpruefen = True
   Else
      MsgBox "Der Preis muss in Währung sein!"
      Felderpruefen = False
      Exit Function
   End If
   If varTypMatrix(4) = CInt(varFeldMatrix(4)) And varTypMatrix(4) > 0 Then
      Felderpruefen = True
   Else
      MsgBox "Die Grundmenge muss eine natürliche Zahl sein!"
```

```
        Felderpruefen = False
        Exit Function
    End If
    If varTypMatrix(5) = CDbl(varFeldMatrix(5)) And varTypMatrix(5) > 0 Then
        Felderpruefen = True
    Else
        MsgBox "Der Faktor muss eine positive Fließkommazahl sein!"
        Felderpruefen = False
        Exit Function
    End If
    If varTypMatrix(6) = CInt(varFeldMatrix(6)) And varTypMatrix(6) > 0 Then
        Felderpruefen = True
    Else
        MsgBox "Die Istmenge muss eine natürliche Zahl sein!"
        Felderpruefen = False
        Exit Function
    End If
    If varTypMatrix(7) = CStr(varFeldMatrix(7)) Then
        Felderpruefen = True
    Else
        MsgBox "Der Firmaname muss eine Zeichenkette(String) sein!"
        Felderpruefen = False
        Exit Function
    End If
    If varTypMatrix(8) = CStr(varFeldMatrix(8)) Then
        Felderpruefen = True
    Else
        MsgBox "Die Straße und Hausnummer muss eine Zeichenkette(String) sein!"
        Felderpruefen = False
        Exit Function
    End If
    If varTypMatrix(9) = CStr(varFeldMatrix(9)) Then
        Felderpruefen = True
    Else
        MsgBox "Die Postleitzahl muss eine Zeichenkette(String) sein!"
        Felderpruefen = False
        Exit Function
    End If
    If varTypMatrix(10) = CStr(varFeldMatrix(10)) Then
        Felderpruefen = True
    Else
        MsgBox "Der Ort muss eine Zeichenkette(String) sein!"
        Felderpruefen = False
        Exit Function
    End If
End Function
```

```
Sub Tabelleleeren()
    Dim varAnzahlZeilen As Integer
    varAnzahlZeilen = Worksheets("tblMKAUS").Cells(10, 1).CurrentRegion.Rows.Count + 10
    Worksheets("tblMKAUS").Rows("11:" & varAnzahlZeilen).Delete Shift:=xlUp
    For i = 3 To 7
        Worksheets("tblMKAUS").Cells(i, 3).Value = ""
        Worksheets("tblMKAUS").Cells(i, 9).Value = ""
    Next i
End Sub
```

```
Sub Kopfteilausgeben()
    Worksheets("tblMKAUS").Cells(7, 3).NumberFormat = "@"
    Worksheets("tblMKAUS").Cells(7, 3).Value = Me.cboAtnr.Text
    Worksheets("tblMKAUS").Cells(3, 3).NumberFormat = "@"
    Worksheets("tblMKAUS").Cells(3, 3).Value = Me.txtFirma.Text
    Worksheets("tblMKAUS").Cells(4, 3).NumberFormat = "@"
    Worksheets("tblMKAUS").Cells(4, 3).Value = Me.txtStrasse.Text
    Worksheets("tblMKAUS").Cells(5, 3).NumberFormat = "@"
    Worksheets("tblMKAUS").Cells(5, 3).Value = Me.txtPlz.Text & " " & Me.txtOrt.Text
    Worksheets("tblMKAUS").Cells(7, 9).Value = Me.txtDatum.Text
End Sub
```

```
Sub Datenausgeben(varDieZeile As Integer)
    Dim varAnfangszeile As Integer
    Dim varGesamtzeile As Integer
    Dim varGesamtMenge As Double, varPlanSumme As Currency, varIstSumme As Currency, _
    varAbweichung As Double, varGesamtAbweichung As Double
    varGesamtMenge = 0
    varPlanSumme = 0
    varIstSumme = 0
    varAbweichung = 0
    varGesamtAbweichung = 0
    varAnfangszeile = 10
    varGesamtzeile = 0
    varGesamtzeile = varAnfangszeile + varDieZeile
    Worksheets("tblMKAUS").Rows(varGesamtzeile + 1).Insert Shift:=xlDown

    Worksheets("tblMKAUS").Cells(varGesamtzeile, 1).NumberFormat = "@"
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 2).NumberFormat = "@"
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 3).NumberFormat = "0.00""€"""
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 4).NumberFormat = "0"
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 5).NumberFormat = "0.0000"
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 6).NumberFormat = "0"
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 7).NumberFormat = "0.000"
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 8).NumberFormat = "0.00""€"""
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 9).NumberFormat = "0.00""€"""
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 10).NumberFormat = "0.00%"

    Worksheets("tblMKAUS").Cells(varGesamtzeile, 2).Value = Me.cboBstn.Text
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 1).Value = Me.txtBsbez.Text
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 3).Value = Me.txtPreis.Text * 1
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 4).Value = Me.txtGrundmenge.Text * 1
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 5).Value = Me.txtFaktor.Text * 1
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 6).Value = Me.txtIstmenge.Text * 1
    varGesamtMenge = Me.txtGrundmenge.Text * Me.txtFaktor.Text
    varPlanSumme = varGesamtMenge * Me.txtPreis.Text
    varIstSumme = Me.txtIstmenge.Text * Me.txtPreis.Text
    varAbweichung = (varPlanSumme - varIstSumme) / varPlanSumme
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 7).Value = varGesamtMenge
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 8).Value = varPlanSumme
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 9).Value = varIstSumme
    Worksheets("tblMKAUS").Cells(varGesamtzeile, 10).Value = varAbweichung
    varGesamtPlan = varGesamtPlan + varPlanSumme
    varGesamtIst = varGesamtIst + varIstSumme
    varGesamtAbweichung = varGesamtPlan - varGesamtIst
    Worksheets("tblMKAUS").Cells(4, 9).Value = varGesamtPlan
    Worksheets("tblMKAUS").Cells(5, 9).Value = varGesamtIst
    Worksheets("tblMKAUS").Cells(6, 9).Value = varGesamtAbweichung
End Sub
```

A4.7) Wie schon im Abschnitt A4.2.2.6 erläutert wurde, muss das Programm als Ergebnis Abb. A4.7a für den Fall „Ausgeben“ liefern.

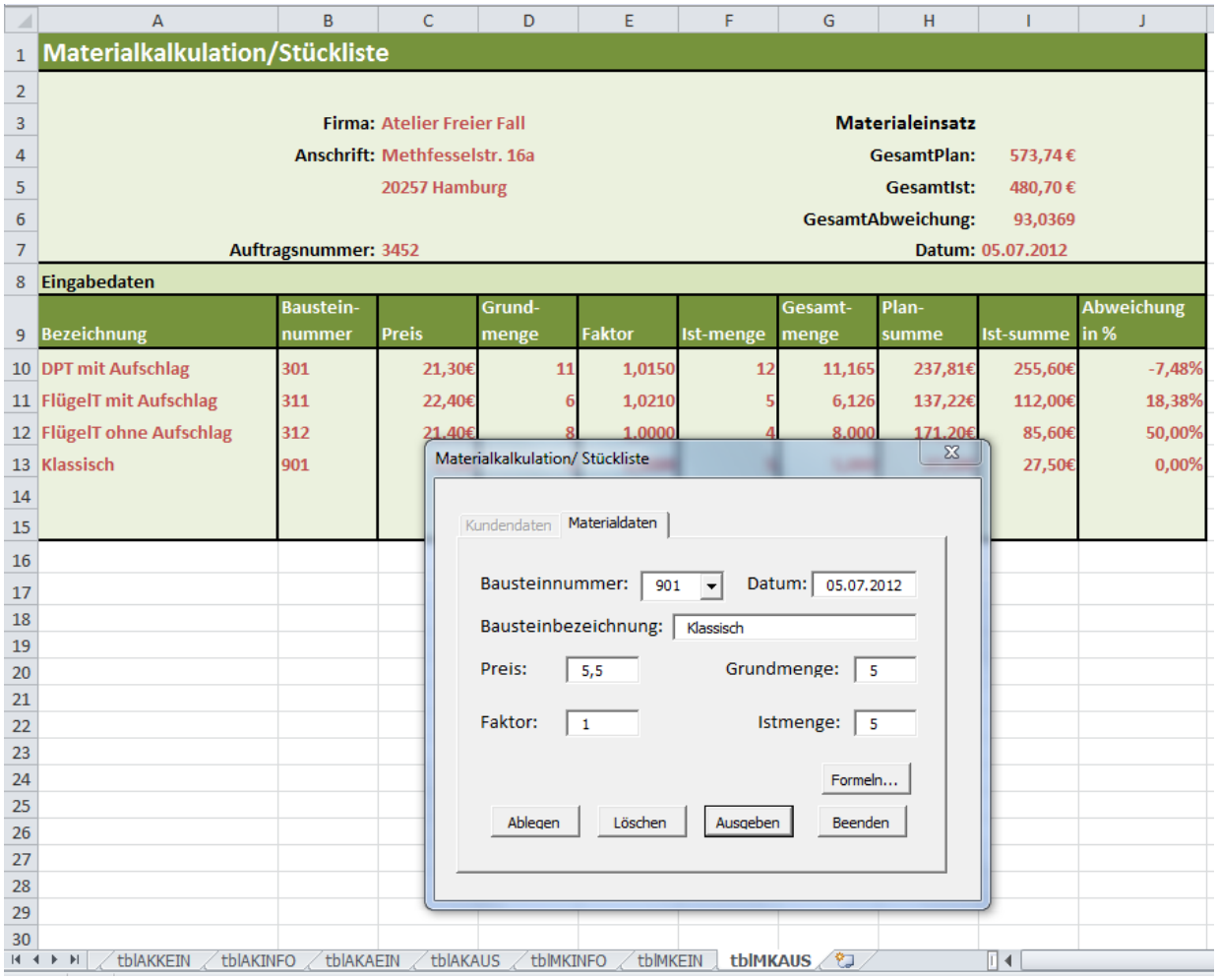


Abb. A4.7a Das Layout beim Ausgeben

A4.8) **LITERATUR**

- Investition und Finanzierung von Ulrich Ermschel et al., ISBN-978-3-7908-2745-3, Springer-Verlag Berlin Heidelberg 2011(https://kataloge.uni-hamburg.de/DB=2/SET=5/TTL=1/SHW?FRST=3)
- Automatisierung und Programmierung mit Excel 2010 (www.rrzn.uni-hannover.de/buecher.html)
- VBA – Programmierung mit Excel von Johannes Gogolok (VBA mit Excel von Uni-Hagen.pdf)
- Grundlagen zu Excel und VBA von Mehdi Bandegani (aik5.pdf und aik6.pdf)

A5) Aufgabe: Eine Datenbank für Auftragskalkulation

Kurzerklärung

Die Definition nach DIN: "Ermittlung der voraussichtlich kostenwirksamen Auftragsleistungen und ihre Bewertung" erscheint bei genauerer Betrachtung als äußerst problembeladen. Sie sagt nämlich nichts darüber aus, wer die Kalkulation durchführt. Gemeinhin werden Kalkulationen von demjenigen erstellt, der auch die Arbeiten durchführt. Hierfür gibt es bereits den Begriff der "Angebotskalkulation". Ein Auftraggeber hingegen führt keine Kalkulation durch, hierfür müsste er ja entstehende Kosten berechnen, sondern eher eine Aufwandsschätzung. In der Regel wird er aber ausschließlich die zu erbringenden Leistungen beschreiben und für diese dann Angebote einholen.

Unter **Kalkulation** versteht man im Allgemeinen

- der Ermittlung der Material-, Maschinen- und Lohnkosten
- der Ermittlung von Brutto- und Netto-Verkaufspreises abhängig von Vertriebskanal und Kundenrabattgruppe.

Dabei kann unterschieden werden zwischen der **Vorkalkulation** in der Planungsphase und der **Nachkalkulation** nach Abschluss aller Produktions- bzw. Handels- und Absatzvorgänge. Die Abweichungen aus Vor- und Nachkalkulation sollten interpretiert werden und ins Kosten-Controlling und die Preisgestaltung zurückfließen.

Beschreibung

Erstellen Sie eine Datenbank, mit der eine Liste der Materialien und deren kalkulaktionsdaten verwaltet und ermittelt werden. In dieser Aufgabestellung geht es u.a darum, mit einer Userform (Abb. A5.2a und Abb. A5.2b) die Kundendaten (Tab. A5.1a) und die Materialdaten (Tab. A5.1b) zu verwalten, sie mit den Kalkulationsformeln (Abb. A5.5a) in eine Ausgabetabelle (Tab. A5.7a) zu übertragen. Ein leerer Entwurf der Ausgabetabelle (Tab. A5.3a) soll ohne Macro-Aufzeichnung erstellt werden.

Wichtige Funktionen
Aufbau einer Exceldatei zu einer automatisierten Auftragskalkulation mit folgenden Funktionen:

- Umgang mit zwei Stammdatentabellen und einer Userform
- Programmierung und Umgang mit den zwei bzw. mehreren Datensätzen
- Konzeption und Realisation der Algorithmen zu der Aufgabestellung
- Umgang mit den benutzerdefinierten Prozeduren
- Berücksichtigung der Qualitätsfaktoren (Korrektheit, Robustheit, benutzerfreunlichkeit usw)

Vorgehensweise

Erstellen Sie eine Exceldatei mit dem Namen axxxxxxx.xLsm (xxxxxxx steht für Ihre Matrikelnummer). In dieser Aufgabe werden drei Tabellen (`tblAKKEIN` für die Kundendaten (Tab. A5.1a) und `tblAKKEIN` für die Auftragskalkulation (Tab. A5.1b) als Stammdatentabellen und `tblAKAUS` als Ausgabetabelle (Tab. A5.7a)) und eine Userform mit dem Namen `frmAuftrag` benötigt. Die Userform (Abb. A5.2a und Abb. A5.2b) muss aus zwei Seiten bestehen, damit jede Seite für die Verwaltung der Datensätze in der entsprechenden Stammdatentabelle dargestellt wird.

A5.1.1) Das Tabellenblatt (Tab. A5.1a) kann `tblAKKEIN` heißen und beinhaltet Datensätze für die `Auftragsnummer` (Spalte A), `Firma` (Spalte B), `Straße` (Spalte C), `Postleitzahl` (Spalte D) und den `Ort` (Spalte E).
A5.1.2) Das Tabellenblatt (Tab. A5.1b) kann `tblAKAEIN` heißen und beinhaltet Datensätze für die `Bausteinbezeichnung` (Spalte A), `Bausteinnummer` (Spalte B), `Menge` (Spalte C) und den `Preis` (Spalte D).

	A	B	C	D	E
1	3452	Atelier Freier Fall	Methfesselstr. 16a	20257	Hamburg
2	3458	Beutin	Jungfernstieg 11/12	20354	Hamburg
3	3481	Beutin	Breite Straße 91	23552	Lübeck
4	3492	Come Clother	Brauerknechtgraben 53	20459	Hamburg
5	3495	Cashmere House	Jägerstraße 61	10117	Berlin
6					
7	**Auftrags-nummer**	**Firma**	**Straße**	**PLZ**	**Ort**

Tab. A5.1a `tblAKKEIN`

	A	B	C	D	E
1	Business1	902	33	6,30 €	
2	City 1	904	34	9,20 €	
3	DPT mit Aufschlag	301	40	21,30 €	
4	FlügelT mit Aufschlag	311	45	22,40 €	
5	FlügelT ohne Aufschlag	312	12	21,40 €	
6	Klassisch	901	30	5,50 €	
7	Vpasse	911	20	9,55 €	
8					
9	**Bausteinbezeichnung**	**Baustein-nummer**	**Menge**	**Preis**	
10					

tblAKAEIN tblAKAUS

Tab. A5.1b `tblAKAEIN`

A5.2) Das Userformblatt (Abb. A5.2a und Abb. A5.2b) kann `frmAuftrag` heißen und besteht aus einem Steuerelement „ Multiseiten“ mit zwei Seiten. Die erste Seite für die Kundendaten und die zweite Seite für die Auftragskalkulation (Materialdaten) sollen vorgesehen werden. Die notwendigen Steuerelemente für jede Seite werden wie folgt zusammengefasst:

Steuerelemente für die Kundendaten

Pos.	Bezeichnung (Caption)	Steuerelement	Name
1	Auftragsnummer:	Beschriftungsfeld	Nicht ändern
2	Keine	Kombinationsfeld	cboAtnr
3	Firma:	Beschriftungsfeld	Nicht ändern
4	Keine	Textfeld	txtFirma
5	Straße + Hausnummer:	Beschriftungsfeld	Nicht ändern
6	Keine	Textfeld	txtStrasse
7	Postleitzahl:	Beschriftungsfeld	Nicht ändern
8	keine	Textfeld	txtPLZ
9	Ort:	Beschriftungsfeld	Nicht ändern
10	keine	Textfeld	txtOrt
11	Ablegen	Befehlsschaltfläche	cmdKAblegen
12	Löschen	Befehlsschaltfläche	cmdKLoeschen
13	Beenden	Befehlsschaltfläche	cmdKBeenden

Steuerelemente für die Materialdaten (Auftragskalkulation)

Pos.	Bezeichnung (Caption)	Steuerelement	Name
1	Bausteinbezeichnung:	Beschriftungsfeld	Nicht ändern
2	Keine	Kombinationsfeld	cboBsbez
3	Bausteinnummer:	Beschriftungsfeld	Nicht ändern
4	Keine	Textfeld	txtBstn
5	Datum:	Beschriftungsfeld	Nicht ändern
6	Keine	Textfeld	txtDatum
7	Preis:	Beschriftungsfeld	Nicht ändern
8	keine	Textfeld	txtPreis
9	Menge:	Beschriftungsfeld	Nicht ändern
10	keine	Textfeld	txtMenge
11	Eingabedaten:	Rahmen	Nicht ändern
12	Auftragsmenge:	Beschriftungsfeld	Nicht ändern
13	keine	Textfeld	txtAuftragsmenge
14	Ablegen	Befehlsschaltfläche	cmdAAblegen
15	Löschen	Befehlsschaltfläche	cmdALoeschen
16	Ausgeben	Befehlsschaltfläche	cmdAAusgeben
17	Forrmeln…	Befehlsschaltfläche	cmdAInfo
18	Beenden	Befehlsschaltfläche	cmdABeenden

Erläuterung zu dem Programmablauf für die Kundendaten

A5.2.1.1) Beim Starten des Userformblatts (Abb. A5.2a) müssen alle Einträge der ersten Spalte von der Tabelle `tblAKKEIN` (Tab. A4.1a) in das Kombinationsfeld `cboAtnr` eingefügt werden.

A5.2.1.2) Wenn ein Eintrag für die Auftragsnummer gewählt wird, muss der entsprechende Datensatz aus der Tabelle `tblAKKEIN` in den Textfeldern erscheinen.

A5.2.1.3) Wenn der Benutzer einen neuen Eintrag in das Userformblatt (Abb. A5.2a) einträgt und auf „Ablegen“ klickt, muss der neue Datensatz in die Tabelle `tblAKKEIN` hinzugefügt werden und anschließend sortiert werden.

A5.2.1.4) Wenn der Benutzer einen Eintrag wählt, der schon in der Tabelle `tblAKKEIN` existiert, muss der Datensatz beim Klicken auf „Ablegen“ ersetzt werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz ersetzt werden soll.)

A5.2.1.5) Wenn der Benutzer einen Eintrag wählt und auf die Befehlsschaltfläche „Löschen“ klickt, muss der Datensatz aus der Tabelle `tblAKKEIN` gelöscht werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz gelöscht werden soll.)

A5.2.1.7) Die Befehlsschaltfläche „Beenden“ blendet die Userform aus.

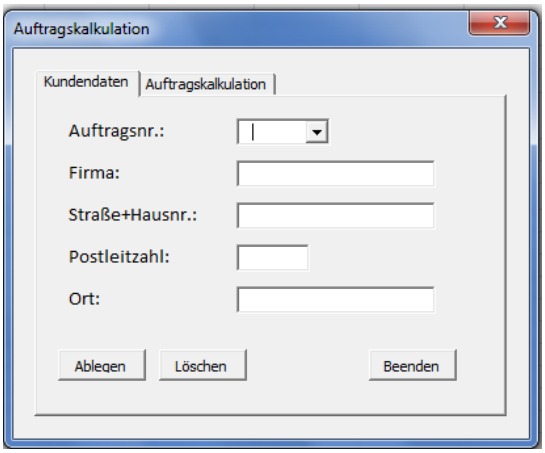


Abb. A5.2a `frmAuftrag` „Kundendaten“

Erläuterung zu dem Programmablauf für die Materialdaten (Auftragskalkulation)

A5.2.2.1) Beim Starten des Userformblatts (Abb. A5.2b) müssen alle Einträge der ersten Spalte von der Tabelle `tblAKAEIN` (Tab. A5.1b) in das Kombinationsfeld `cboBsbez` eingefügt werden.

A5.2.2.2) Wenn ein Eintrag für die Bausteinbezeichnung gewählt wird, muss der entsprechende Datensatz aus der Tabelle `tblAKAEIN` in den Textfeldern erscheinen.

A5.2.2.3) Wenn der Benutzer einen neuen Eintrag in das Userformblatt (Abb. A5.2b) einträgt und auf „Ablegen“ klickt, muss der neue Datensatz in die Tabelle `tblMKEIN` hinzugefügt werden und anschließend sortiert werden.

A5.2.2.4) Wenn der Benutzer einen Eintrag wählt, der schon in der Tabelle `tblAKAEIN` existiert, muss der Datensatz beim Klicken auf „Ablegen“ ersetzt werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz ersetzt werden soll.)

A5.2.2.5) Wenn der Benutzer einen Eintrag wählt und auf die Befehlsschaltfläche „Löschen“ klickt, muss der Datensatz aus der Tabelle `tblAKAEIN` gelöscht werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, ob der Datensatz gelöscht werden soll.)

A5.2.2.6) Wenn der Benutzer auf die Befehlsschaltfläche „Ausgeben“ klickt, müssen alle in der Userform `frmAuftrag` ausgewälten Daten und die eingegebenen Auftragsmenge mit den vorgegebenen Formeln (Abb. A5.5a) berechnet und in die Ausgabetabelle(Tab. A5.3a) `tblAKAUS` übertragen werden. Hierbei sollen insbesonders die Softwarequalitäten berücksichtigt werden. Z.B. sollen Sicherheitsmeldungen ausgegeben, wenn der Benutzer keinen Eintrag gewählt hat, wenn die Inhalte der Textfelder ungültige Datentypen beinhalten.

A5.2.2.7) Die Befehlsschaltfläche „Beenden“

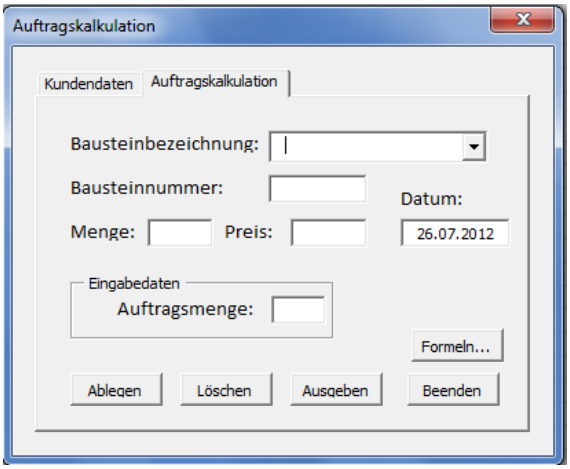


Abb. A5.2b `frmAuftrag` „`Auftragskalkulation`"

blendet die Userform aus.

A5.3) Erstellen Sie eine Tabelle (Tab. A5.3a) mit dem Namen `tblAKAUS`. Die Tabelle soll folgende Merkmale haben:

- Die Spaltenbreiten (A=40 Pixel, B=145 Pixel, C bis E=75 Pixel und G-H=80 Pixel) und die Zeilenhöhen (Zeile1=32 Pixel und Zeile2-34=25 Pixel).
- Die Hintergrundfarben von den Zellen A1-H1 sind Dunkelblau, Text2, heller 40%, von den Zellen A2-H5 sind Blau, Akzent 1, heller 80%, von den Zellen A15-H15, A21-H22, A25-H25, A28-H28, A32-H32 und A34-H34 sind Olivgrün, Akzent 3, heller 80%.
- Der äußere Rahmen und die vertikal und horizontal inneren Rahmen haben die Breite von 1 ½ Pt und die Farbe schwarz, Text 1, heller 5%.
- Die Zelle A1 hat einen Schriftgrad 16, eine Schriftfabe weiß, Hintergrund 1, dunkler 5%. Die Zellen B11-E11 und B17-E18 haben einen Schriftgrad von 10. Alle anderen Zellen haben einen Schriftgrad von 12. Für alle Zellen soll eine Schriftart Calibri gewählt werden.
- Es wird dauf hingewiesen, dass die Zellenformatierungen für die ausgabedaten in Codes vorgenommen werden müssen.

	A	B	C	D	E	F	G	H
1	**Auftragskalkulation**							
2		**Firma:**						
3		**Anschrift**						
4								
5		**Auftrags-Nr:**				**Datum:**	06.07.2012	
6						Auftragsmenge:		
7						**Vorkalkulation**		
8	Pos.	Bezeichnung				GKSatz	Kosten	
9						%	Stück	Auftrag
10	1	**Material**						
11		Bausteinbezeichnung	BausteinNr.	Menge	Preis			
12								
13								
14	2	**Materialgemeinkosten**				**11,0%**		
15	3	**MATERIALKOSTEN**						
16	4	**Fertigungskosten**						
17		Std	Ko-St.		Std_Satz			
18	4.1	56,00	F1		16,80	**195,00%**		
19								
20	5	Sondereinzelkosten der Fertigung						
21	6	**SUMME FERTIGUNGSKOSTEN**						
22	7	**HERSTELLKOSTEN:**						
23	8	Verwaltungsgemeinkosten				**10,50%**		
24	9	Vertriebsgemeinkosten				**19,50%**		
25	10	**SELBSTKOSTEN:**						
26	11	Kalk. Gewinn %:				**12,5%**		
27	12	Provision				**3,0%**		
28	13	**NETTO-PREIS:**						
29	14	Frachtkosten des Auftrags				**100**		
30	15	Verpackungskosten des Auftrags				**300**		
31	16	Skonto				**2,0%**		
32	17	**ANGEBOTSPREIS (netto):**						
33	18	MwSt				**16,0%**		
34	19	**ANGEBOTSPREIS (brutto):**						
35								

Tab. A5.3a `tblAKAUS`

Abb. A5.5a**) Formeln und Erklärung:**

Stück	=	Menge * Preis
Auftrag	=	Stück * Auftragsmenge
Materialgemeinkosten pro Stück	=	Summe aller Materialien * GKSatz
Materialgemeinkosten pro Auftrag	=	Materialgemeinkosten pro Stück * Auftragsmenge
Materialkosten pro Stück	=	Summe aller Materialien + Materialgemeinkosten pro Stück
Materialkosten pro Auftrag	=	Materialkosten pro Stück * Auftragsmenge
Fertigungskosten pro Stück	=	Std * Std_Satz *(1+ GKSatz)
Fertigungskosten pro Auftrag	=	Fertigungskosten pro Stück * Auftragsmenge
Sondereinzelkosten der Fertigung	=	festgelegt durch Fertigungseinrichtungen
Summe Fertigungskosten pro Stück	=	Summe aller Fertigungskosten pro Stück
Summe Fertigungskosten pro Auftrag	=	Summe Fertigungskosten pro Stück * Auftragsmenge
Herstellkosten pro Stück	=	Materialkosten + Summe Fertigungskosten
Herstellkosten pro Auftrag	=	Herstellkosten pro Stück * Auftragsmenge
Verwaltungsgemeinkosten	=	Herstellkosten * GKSatz
Vertriebsgemeinkosten	=	Herstellkosten * GKSatz
Selbstkosten pro Stück	=	Herstellkosten + Verwalt.kosten + Vertriebsgemeinkosten
Selbstkosten pro Auftrag	=	Selbstkosten pro Stück * Auftragsmenge
Kalk. Gewinn	=	Selbstkosten pro Stück * GKSatz
Provision	=	(Selbstkosten pro Stück + Kalk. Gewinn) * GKSatz / (1 - GKSatz)
Netto-Preis pro Stück	=	Selbstkosten pro Stück + Kalk. Gewinn + Provision
Netto-Preis pro Auftrag	=	Netto-Preis pro Stück * Auftragsmenge
Frachtkosten	=	GKSatz / Auftragsmenge
Verpackungskosten	=	GKSatz / Auftragsmenge
Skonto	=	(Netto-Preis pro Stück + Frachtkosten + Verpackungskosten) * GKSatz / (1-GKSatz)
Angebotspreis(netto) pro Stück	=	Netto-Preis pro Stück + Verpackungskosten + Frachtkosten + Skonto
Angebotspreis(netto) pro Auftrag	=	Angebotspreis pro Stück * Auftragsmenge
Mwst pro Stück	=	Angebotspreis pro Stück * GKSatz
Mwst pro Auftrag	=	Mwst pro Stück * Auftragsmenge
Angebotspreis(brutto) pro Stück	=	Angebotspreis(netto) pro Stück + Mwst. pro Stück
Angebotspreis(brutto) pro Auftrag	=	Angebotspreis(brutto) pro Stück * Auftragsmenge

Programmablaufplan für die Kundendaten

Der Programmablaufplan (Abb. A5.6a) bezieht sich auf die Abschnitten A5.2.1.1 – A5.2.1.7 Entwickeln und realisieren Sie bitte einen Programmablaufplan und ein Programmcode, in dem kurze Codes und vereinfachte Aktionen für den Benutzer ermöglicht werden.

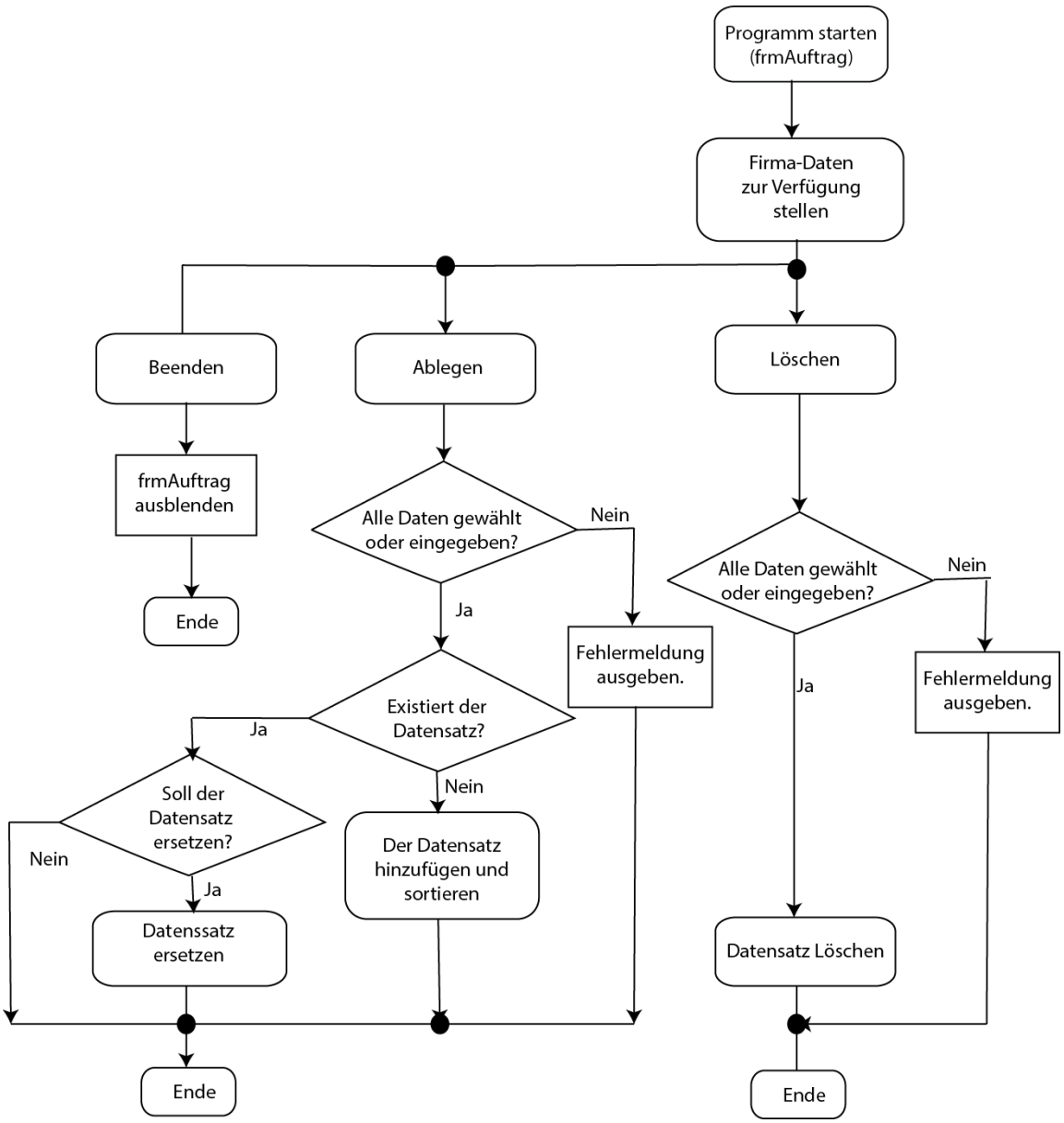


Abb. A5.6a Programmablaufplan zu der kundendaten

Programmablaufplan für die Kundendaten

Der Programmablaufplan (Abb. A5.6b) bezieht sich auf die Abschnitten A5.2.2.1 – A5.2.2.7 Entwickeln und realisieren Sie bitte einen Programmablaufplan und ein Programmcode, in dem kurze Codes und vereinfachte Aktionen für den Benutzer ermöglicht werden.

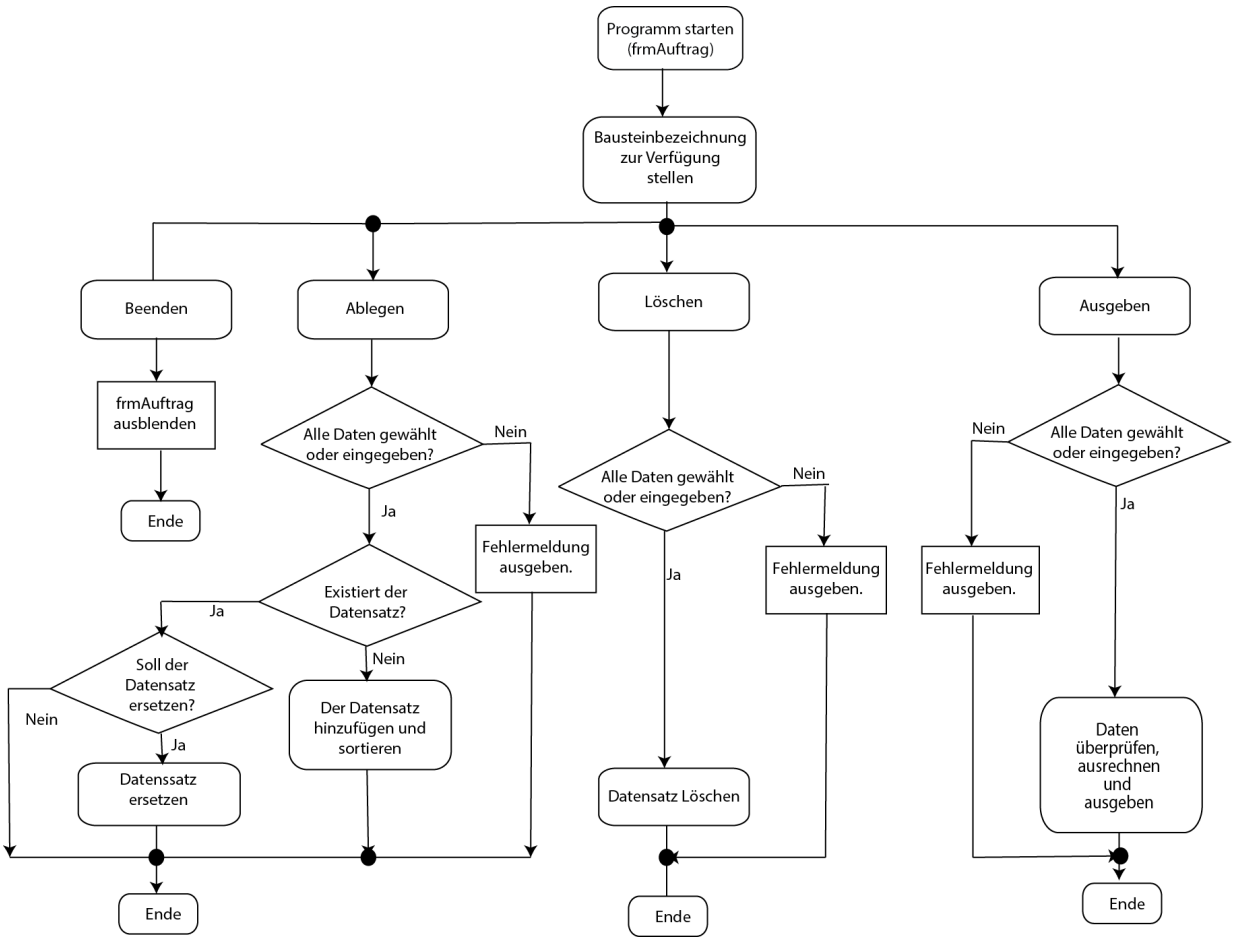


Abb. A5.6b Programmablaufplan zu dem Auftrag

A5.6)Folgende VBA-Codes dienen dazu, um die in den Abschnitten A5.2.1.1-A5.2.1.7 und A5.2.2.1-A5.2.2.7 Programmablaufpläne zu implementieren:

```
Dim varKWelcherEintrag As Integer  'Eintrag für Auftragsnummer
Dim varNeuerEintrag As Integer
Dim varAWelcherEintrag As Integer  'Eintrag für varBausteinbezeichnung
Dim varEinzelkosten As Double
Dim varMaterialkosten As Double
```

```
Private Sub UserForm_Initialize()
   varEinzelkosten = 0
   varMaterialkosten = 0
   varNeuerEintrag = 0
   KMaskeloeschen
   KEinfuegen
   AMaskeloeschen
   AEinfuegen
   Tabelleleeren
   Me.txtDatum.Text = Date
   MultiPage1.Pages(0).Enabled = True
End Sub
```

```
Sub KEinfuegen()
   Dim varAuftragsnummer As Integer
   varAuftragsnummer = Worksheets("tblAKKEIN").Cells(1, 1).CurrentRegion.Rows.Count
```

```
    For i = 1 To varAuftragsnummer
        Me.cboAtnr.AddItem Worksheets("tblAKKEIN").Cells(i, 1).Text
    Next i
End Sub
Sub AEinfuegen()
    Dim varBausteinbezeichnung As Integer
    varBausteinbezeichnung = Worksheets("tblAKAEIN").Cells(1, 1).CurrentRegion.Rows.Count
    For i = 1 To varBausteinbezeichnung
        Me.cboBsbez.AddItem Worksheets("tblAKAEIN").Cells(i, 1).Text
    Next i
End Sub
Sub KMaskeloeschen()
    Me.cboAtnr.Clear
    Me.cboAtnr.Text = ""
    Me.txtFirma.Text = ""
    Me.txtStrasse.Text = ""
    Me.txtPlz.Text = ""
    Me.txtOrt.Text = ""
End Sub
Sub AMaskeloeschen()
    Me.cboBsbez.Clear
    Me.cboBsbez.Text = ""
    Me.txtBstn.Text = ""
    Me.txtPreis.Text = ""
    Me.txtMenge.Text = ""
    Me.txtAuftragsmenge.Text = ""
End Sub
Private Sub cmdKBeenden_Click()
    varNeuerEintrag = 0
    ActiveWindow.WindowState = xlNormal
    Me.Hide
End Sub
Private Sub cmdABeenden_Click()
    varNeuerEintrag = 0
    ActiveWindow.WindowState = xlNormal
    Me.Hide
End Sub
Private Sub cboAtnr_Click()
    varKWelcherEintrag = Me.cboAtnr.ListIndex + 1
    Me.txtFirma.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 2).Value
    Me.txtStrasse.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 3).Value
    Me.txtPlz.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 4).Value
    Me.txtOrt.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 5).Value
End Su
Private Sub cboAtnr_Enter()
    Me.txtFirma.Text = ""
    Me.txtStrasse.Text = ""
    Me.txtPlz.Text = ""
    Me.txtOrt.Text = ""
End Sub
Private Sub cboAtnr_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Me.txtFirma.Text = ""
    Me.txtStrasse.Text = ""
    Me.txtPlz.Text = ""
    Me.txtOrt.Text = ""
End Sub
Private Sub cboBsbez_Click()
    varAWelcherEintrag = Me.cboBsbez.ListIndex + 1
    Me.txtBstn.Text = Worksheets("tblAKAEIN").Cells(varAWelcherEintrag, 2).Value
    Me.txtMenge.Text = Worksheets("tblAKAEIN").Cells(varAWelcherEintrag, 3).Value
    Me.txtPreis.Text = Worksheets("tblAKAEIN").Cells(varAWelcherEintrag, 4).Value
End Sub

Private Sub cboBsbez_Enter()
    Me.txtBstn.Text = ""
```

```
    Me.txtMenge.Text = ""
    Me.txtPreis.Text = ""
End Sub
Private Sub cboBsbez_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Me.txtBstn.Text = ""
    Me.txtMenge.Text = ""
    Me.txtPreis.Text = ""
End Sub
Private Sub cmdKLoeschen_Click()
    If Me.cboAtnr.Text = "" Then
        Beep
        MsgBox "Sie müssen einen Eintrag wählen."
    Else
        Mldg = "Wollen Sie wirklich diesen Datensatz loeschen ?"
        Stil = vbYesNo + vbCritical + vbDefaultButton2
        Title = "Daten loeschen"
        Ergebnis = MsgBox(Mldg, Stil, Title)
        If Ergebnis = vbYes Then
            Worksheets("tblAKKEIN").Rows(varKWelcherEintrag).Delete Shift:=xlUp
            Worksheets("tblAKKEIN").Activate
            KMaskeloeschen
            KEinfuegen
        End If
    End If
End Sub
Private Sub cmdAloeschen_Click()
    If Me.cboBsbez.Text = "" Then
        Beep
        MsgBox "Sie müssen einen Eintrag wählen."
    Else
        Mldg = "Wollen Sie wirklich diesen Datensatz loeschen ?"
        Stil = vbYesNo + vbCritical + vbDefaultButton2
        Title = "Daten loeschen"
        Ergebnis = MsgBox(Mldg, Stil, Title)
        If Ergebnis = vbYes Then
            Worksheets("tblAKAEIN").Rows(varAWelcherEintrag).Delete Shift:=xlUp
            Worksheets("tblAKAEIN").Activate
            AMaskeloeschen
            AEinfuegen
        End If
    End If
End Sub
Private Sub cmdkAblegen_Click()
    If Me.cboAtnr.Text = "" Then
        MsgBox "Sie müssen einen Eintrag wählen."
    Else
        If Me.cboAtnr.Text > "" And varKWelcherEintrag > 0 Then
            If Me.cboAtnr.Text = Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 1).Value Then
                Mldg = "Wollen Sie diesen datensatz Ersetzen ?"
                Stil = vbYesNo + vbCritical + vbDefaultButton2
                Title = "Kundendaten"
                Ergebnis = MsgBox(Mldg, Stil, Title)
                If Ergebnis = vbYes Then
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 1).Value = Me.cboAtnr.Text
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 2).Value = Me.txtFirma.Text
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 3).Value = Me.txtStrasse.Text
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 4).Value = Me.txtPlz.Text
                    Worksheets("tblAKKEIN").Cells(varKWelcherEintrag, 5).Value = Me.txtOrt.Text
                    Worksheets("tblAKKEIN").Activate
                    KMaskeloeschen
                    KEinfuegen
                    Exit Sub
                End If
            Else
                Worksheets("tblAKKEIN").Activate
                KNeueintragen
```

```
            KMaskeloeschen
            KEinfuegen
          End If
        ElseIf Me.cboAtnr.Text > "" And varKWelcherEintrag = 0 Then
          Worksheets("tblAKKEIN").Activate
          KNeueintragen
          KMaskeloeschen
          KEinfuegen
        Else
          MsgBox "Dieser Zustand kommt gar nicht vor!"
          Exit Sub
        End If
    End If
End Sub
```

```
Private Sub cmdAAblegen_Click()
    If Me.cboBsbez.Text = "" Then
        MsgBox "Sie müssen einen Eintrag wählen."
    Else
        If Me.cboBsbez.Text > "" And varAWelcherEintrag > 0 Then
            If Me.cboBsbez.Text = Worksheets("tblAKAEIN").Cells(varAWelcherEintrag, 1).Value Then
                Mldg = "Wollen Sie die alte Daten Ersetzen ?"
                Stil = vbYesNo + vbCritical + vbDefaultButton2
                Title = "Auftragsdaten"
                Ergebnis = MsgBox(Mldg, Stil, Title)
                If Ergebnis = vbYes Then
                    Worksheets("tblAKAEIN").Cells(varAWelcherEintrag, 1).Value = Me.cboBsbez.Text
                    Worksheets("tblAKAEIN").Cells(varAWelcherEintrag, 2).Value = Me.txtBstn.Text
                    Worksheets("tblAKAEIN").Cells(varAWelcherEintrag, 3).Value = Me.txtMenge.Text
                    Worksheets("tblAKAEIN").Cells(varAWelcherEintrag, 4).Value = Me.txtPreis.Text
                    Worksheets("tblAKAEIN").Activate
                    AMaskeloeschen
                    AEinfuegen
                    Exit Sub
                End If
            Else
                Worksheets("tblAKAEIN").Activate
                ANeueintragen
                AMaskeloeschen
                AEinfuegen
            End If
        ElseIf Me.cboBsbez.Text > "" And varAWelcherEintrag = 0 Then
            Worksheets("tblAKAEIN").Activate
            ANeueintragen
            AMaskeloeschen
            AEinfuegen
        Else
            MsgBox "Dieser Zustand kommt nicht vor!"
            Exit Sub
        End If
    End If
End Sub
```

```
Sub KNeueintragen()
    Dim varNeueZeile As Integer
    varNeueZeile = Worksheets("tblAKKEIN").Cells(1, 1).CurrentRegion.Rows.Count + 1
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 1).NumberFormat = "@"
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 1).Value = Me.cboAtnr.Text
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 2).NumberFormat = "@"
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 2).Value = Me.txtFirma.Text
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 3).NumberFormat = "@"
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 3).Value = Me.txtStrasse.Text
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 4).NumberFormat = "@"
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 4).Value = Me.txtPlz.Text
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 4).NumberFormat = "@"
    Worksheets("tblAKKEIN").Cells(varNeueZeile, 5).Value = Me.txtOrt.Text
    With Worksheets("tblAKKEIN")
        .Range(.Cells(1, 1), .Cells(varNeueZeile, 5)).Sort Key1:= _
```

```
        Worksheets("tblAKKEIN").Columns("A"), Header:=xlGuess
    End With
End Sub
Sub ANeueintragen()
    Dim varNeueZeile As Integer
    varNeueZeile = Worksheets("tblAKAEIN ").Cells(1, 1).CurrentRegion.Rows.Count + 1
    Worksheets("tblAKAEIN ").Cells(varNeueZeile, 1).Value.NumberFormat = "@"
    Worksheets("tblAKAEIN ").Cells(varNeueZeile, 1).Value = Me.cboBsbez.Text
    Worksheets("tblAKAEIN ").Cells(varNeueZeile, 2).Value.NumberFormat = "@"
    Worksheets("tblAKAEIN ").Cells(varNeueZeile, 2).Value = Me.txtBsBstn.Text
    Worksheets("tblAKAEIN ").Cells(varNeueZeile, 3).Value.NumberFormat =  "0"
    Worksheets("tblAKAEIN ").Cells(varNeueZeile, 3).Value = Me.txtmenge.Text
    Worksheets("tblAKAEIN ").Cells(varNeueZeile, 4).Value.NumberFormat ="0.00""€"""
    Worksheets("tblAKAEIN ").Cells(varNeueZeile, 4).Value = Me.txtPreis.Text
    With Worksheets("tblAKAEIN ")
        .Range(.Cells(1, 1), .Cells(varNeueZeile, 4)).Sort Key1:= _
            Worksheets("tblAKAEIN ").Columns("A"), Header:=xlGuess
    End With
End Sub
Private Sub cmdAAusgeben_Click()
    If Me.cboBsbez.Text = "" And Me.cboAtnr.Text = "" Then
        Beep
        MsgBox "Sie müssen einen Eintrag wählen."
        Exit Sub
    End If

    If Me.txtAuftragsmenge.Text = "" Then
        MsgBox "Geben Sie die Auftragsmenge an!"
        Exit Sub
    End If
    If Me.cboBsbez.Text > "" And Me.cboAtnr.Text > "" Then
        Worksheets("tblAKAUS").Activate
        Rem die Seite "Kundendaten" ausblenden"
        MultiPage1.Pages(0).Enabled = False
        Worksheets("tblAKAUS").Cells(5, 3).NumberFormat = "@"
        Worksheets("tblAKAUS").Cells(5, 3).Value = Me.cboAtnr.Text
        Worksheets("tblAKAUS").Cells(2, 3).NumberFormat = "@"
        Worksheets("tblAKAUS").Cells(2, 3).Value = Me.txtFirma.Text
        Worksheets("tblAKAUS").Cells(3, 3).NumberFormat = "@"
        Worksheets("tblAKAUS").Cells(3, 3).Value = Me.txtStrasse.Text
        Worksheets("tblAKAUS").Cells(4, 3).NumberFormat = "@"
        Worksheets("tblAKAUS").Cells(4, 3).Value = Me.txtPlz.Text & " " & Me.txtOrt.Text
        Worksheets("tblAKAUS").Cells(6, 7).NumberFormat = "0"
        Worksheets("tblAKAUS").Cells(6, 7).Value = Me.txtAuftragsmenge.Text
        Worksheets("tblAKAUS").Cells(5, 7).Value = Me.txtDatum.Text
        Dim Anfangszeile As Integer
        Dim Gesamtzeilen As Integer
        Anfangszeile = 11
        varNeuerEintrag = varNeuerEintrag + 1
        Gesamtzeilen = Anfangszeile + varNeuerEintrag
        Worksheets("tblAKAUS").Rows(Gesamtzeilen).Insert Shift:=xlDown
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 2).NumberFormat = "@"
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 2).Value = Me.cboBsbez.Text
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 3).NumberFormat = "@"
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 3).Value = Me.txtBstn.Text
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 4).NumberFormat = "0"
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 4).Value = Me.txtMenge.Text
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 5).NumberFormat = "0.00"
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 5).Value = Me.txtPreis.Text
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 7).NumberFormat = "0.00"
        Rem Punkt 1 Material
        varEinzelkosten = Me.txtPreis.Text * Me.txtMenge.Text
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 7).Value = varEinzelkosten
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 8).NumberFormat = "0.00"
        Worksheets("tblAKAUS").Cells(Gesamtzeilen, 8).Value = varEinzelkosten * Me.txtAuftragsmenge.Text
        Rem Punkt 2 Materialsgemeinkosten
```

```
varMaterialkosten = varMaterialkosten + varEinzelkosten
Dim varMaterialGKSatz As Double
varMaterialGKSatz = 0
varMaterialGKSatz = Worksheets("tblAKAUS").Cells(Gesamtzeilen + 3, 6).Value
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 3, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 3, 7).Value = varMaterialkosten * varMaterialGKSatz
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 3, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 3, 8).Value = _
   varMaterialkosten * varMaterialGKSatz * Me.txtAuftragsmenge.Text
Rem Punkt 3 Materialskosten
Dim varSummeMaterialkosten As Double
varSummeMaterialkosten = 0
varSummeMaterialkosten = varMaterialkosten * (1 + varMaterialGKSatz)
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 4, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 4, 7).Value = varSummeMaterialkosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 4, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 4, 8).Value =  varSummeMaterialkosten * Me.txtAuftragsmenge.Text
Rem Punkt 4.1 Fertigungskosten
Dim varFertigungskosten As Double
varFertigungskosten = 0
varFertigungskosten = Worksheets("tblAKAUS").Cells(Gesamtzeilen + 7, 2).Value _
   * Worksheets("tblAKAUS").Cells(Gesamtzeilen + 7, 5).Value _
   * (1 + Worksheets("tblAKAUS").Cells(Gesamtzeilen + 7, 6).Value)
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 7, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 7, 7).Value = varFertigungskosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 7, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 7, 8).Value = varFertigungskosten * Me.txtAuftragsmenge.Text
Rem Punkt 5 Sondereinzelkosten der Fertigung
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 9, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 9, 7).Value = 1000
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 9, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 9, 8).Value = 1000 * Me.txtAuftragsmenge.Text
Rem Punkt 6 Summe Fertigungskosten
Dim varSummeFertigungskosten As Double
varSummeFertigungskosten = 0
varSummeFertigungskosten = 1000 + varFertigungskosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 10, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 10, 7).Value = varSummeFertigungskosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 10, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 10, 8).Value = varSummeFertigungskosten * Me.txtAuftragsmenge.Text
Rem Punkt 7 Herstellkosten
Dim varHerstellkosten As Double
varHerstellkosten = 0
varHerstellkosten = varSummeFertigungskosten + varSummeMaterialkosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 11, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 11, 7).Value = varHerstellkosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 11, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 11, 8).Value = varHerstellkosten * Me.txtAuftragsmenge.Text
Rem Punkt 8 Verwaltungsgemeinkosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 12, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 12, 7).Value = varHerstellkosten * _
   Worksheets("tblAKAUS").Cells(Gesamtzeilen + 12, 6).Value
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 12, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 12, 8).Value = varHerstellkosten * _
   Worksheets("tblAKAUS").Cells(Gesamtzeilen + 12, 6).Value * Me.txtAuftragsmenge.Text
Rem Punkt 9 Vertriebsgemeinkosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 13, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 13, 7).Value = varHerstellkosten * _
   Worksheets("tblAKAUS").Cells(Gesamtzeilen + 13, 6).Value
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 13, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 13, 8).Value = varHerstellkosten * _
   Worksheets("tblAKAUS").Cells(Gesamtzeilen + 13, 6).Value * Me.txtAuftragsmenge.Text
Rem Punkt 10 Selbstkosten
Dim varSelbstkosten As Double
varSelbstkosten = 0
varSelbstkosten = varHerstellkosten + Worksheets("tblAKAUS").Cells(Gesamtzeilen + 12, 7).Value _
```

```
    + Worksheets("tblAKAUS").Cells(Gesamtzeilen + 13, 7).Value
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 14, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 14, 7).Value = varSelbstkosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 14, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 14, 8).Value = varSelbstkosten * Me.txtAuftragsmenge.Text
Rem Punkt 11 Kalk. Gewinn
Dim varKalkgewinn  As Double
varKalkgewinn = 0
varKalkgewinn = varSelbstkosten * Worksheets("tblAKAUS").Cells(Gesamtzeilen + 15, 6).Value
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 15, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 15, 7).Value = varKalkgewinn
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 15, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 15, 8).Value = varKalkgewinn * Me.txtAuftragsmenge.Text
Rem Punkt 12 Provision
Dim varProvision  As Double
varProvision = 0
varProvision = (varSelbstkosten + varKalkgewinn) * _
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 16, 6).Value / _
(1 - Worksheets("tblAKAUS").Cells(Gesamtzeilen + 16, 6).Value)
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 16, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 16, 7).Value = varProvision
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 16, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 16, 8).Value = varProvision * Me.txtAuftragsmenge.Text
Rem Punkt 13 Nettopreis
Dim varNettopreis  As Double
varNettopreis = 0
varNettopreis = varSelbstkosten + varProvision + varKalkgewinn
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 17, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 17, 7).Value = varNettopreis
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 17, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 17, 8).Value = varNettopreis * Me.txtAuftragsmenge.Text
Rem Punkt 14 Frachtkosten
Dim varFrachtkosten  As Double
varFrachtkosten = 0
varFrachtkosten = Worksheets("tblAKAUS").Cells(Gesamtzeilen + 18, 6).Value / Me.txtAuftragsmenge.Text
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 18, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 18, 7).Value = varFrachtkosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 18, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 18, 8).Value = varFrachtkosten * Me.txtAuftragsmenge.Text
Rem Punkt 15 Verpackungskosten
Dim varVerpackungskosten  As Double
varVerpackungskosten = 0
varVerpackungskosten = Worksheets("tblAKAUS").Cells(Gesamtzeilen + 19, 6).Value / Me.txtAuftragsmenge.Text
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 19, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 19, 7).Value = varVerpackungskosten
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 19, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 19, 8).Value = varVerpackungskosten * Me.txtAuftragsmenge.Text
Rem Punkt 16 Skonto
Dim varSkonto  As Double
varSkonto = 0
varSkonto = (varNettopreis + varFrachtkosten + varVerpackungskosten) * _
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 20, 6).Value / (1 - Worksheets("tblAKAUS").Cells(Gesamtzeilen + 20, 6).Value)
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 20, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 20, 7).Value = varSkonto
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 20, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 20, 8).Value = varSkonto * Me.txtAuftragsmenge.Text
Rem Punkt 17 Angebotspreis
Dim varAngebotspreis  As Double
varAngebotspreis = 0
varAngebotspreis = varNettopreis + varFrachtkosten + varVerpackungskosten + varSkonto
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 21, 7).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 21, 7).Value = varAngebotspreis
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 21, 8).NumberFormat = "0.00"
Worksheets("tblAKAUS").Cells(Gesamtzeilen + 21, 8).Value = varAngebotspreis * Me.txtAuftragsmenge.Text
Rem Punkt 18 MWst (Umsatzsteuer)
Dim varMWst  As Double
```

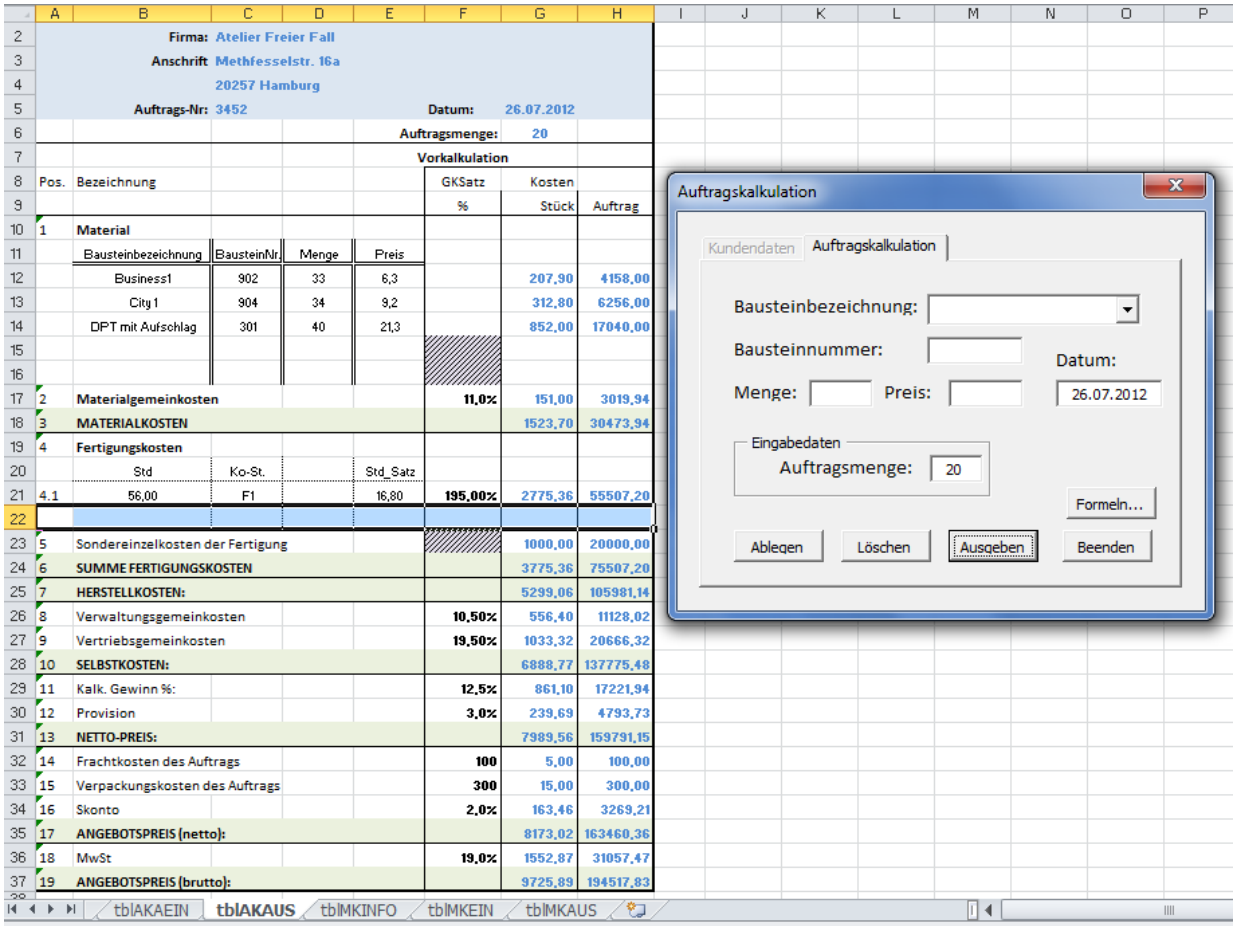
```
        varMWst = 0
        varMWst = varAngebotspreis * Worksheets("tblAKAUS").Cells(Gesamtzeilen + 22, 6).Value
        Worksheets("tblAKAUS").Cells(Gesamtzeilen + 22, 7).NumberFormat = "0.00"
        Worksheets("tblAKAUS").Cells(Gesamtzeilen + 22, 7).Value = varMWst
        Worksheets("tblAKAUS").Cells(Gesamtzeilen + 22, 8).NumberFormat = "0.00"
        Worksheets("tblAKAUS").Cells(Gesamtzeilen + 22, 8).Value = varMWst * Me.txtAuftragsmenge.Text
        Rem Punkt 19 Gesamtangebot
        Dim varGesamtangebot  As Double
        varGesamtangebot = 0
        varGesamtangebot = varAngebotspreis + varMWst
        Worksheets("tblAKAUS").Cells(Gesamtzeilen + 23, 7).NumberFormat = "0.00"
        Worksheets("tblAKAUS").Cells(Gesamtzeilen + 23, 7).Value = varGesamtangebot
        Worksheets("tblAKAUS").Cells(Gesamtzeilen + 23, 8).NumberFormat = "0.00"
        Worksheets("tblAKAUS").Cells(Gesamtzeilen + 23, 8).Value = varGesamtangebot * Me.txtAuftragsmenge.Text
        Me.cboBsbez.Clear

        Me.cboBsbez.Text = ""
        Me.txtBstn.Text = ""
        Me.txtPreis.Text = ""
        Me.txtMenge.Text = ""
        AEinfuegen
    End If
End Sub
```

```
Sub Tabelleleeren()
    Dim varZelleninhalt
    varZelleninhalt = Worksheets("tblAKAUS").Cells(12, 2).Value
    If varZelleninhalt > "" Then
        Dim AnzahlDerZeilen As Integer
        AnzahlDerZeilen = Worksheets("tblAKAUS").Cells(12, 2).CurrentRegion.Rows.Count
        Mldg = "Wollen Sie die alte Daten loeschen?"
        Stil = vbYesNo + vbCritical + vbDefaultButton2
        Title = "Daten loeschen"
        Ergebnis = MsgBox(Mldg, Stil, Title)
        If Ergebnis = vbYes Then
            Worksheets("tblAKAUS").Rows("12:" & AnzahlDerZeilen).Delete Shift:=xlUp
            Worksheets("tblAKAUS").Cells(6, 7).Value = ""
            i = 0
            For i = 2 To 5
                Worksheets("tblAKAUS").Cells(i, 3).Value = ""
            Next i
            Worksheets("tblAKAUS").Cells(5, 7).Value = ""
            Worksheets("tblAKAUS").Cells(6, 7).Value = ""
            For i = 12 To 34
                Worksheets("tblAKAUS").Cells(i, 7).Value = ""
                Worksheets("tblAKAUS").Cells(i, 8).Value = ""
            Next i
        End If
    Else
        MsgBox "Ihre Tabelle ist leer"
    End If
End Sub
```

```
Private Sub cmdAInfo_Click()
  Worksheets("tblAKINFO").Activate
End Sub
```

A5.7) Wie schon im Abschnitt A5.2.2.6 erläutert wurde, muss das Programm als Ergebnis Abb. A5.7a für den Fall „Ausgeben“ liefern.



		Firma:	Atelier Freier Fall					
		Anschrift	Methfesselstr. 16a					
			20257 Hamburg					
		Auftrags-Nr:	3452			Datum:	26.07.2012	
						Auftragsmenge:	20	
						Vorkalkulation		
Pos.	Bezeichnung					GKSatz	Kosten	
						%	Stück	Auftrag
1	Material							
	Bausteinbezeichnung	BausteinNr.	Menge	Preis				
	Business1	902	33	6,3			207,90	4158,00
	City1	904	34	9,2			312,80	6256,00
	DPT mit Aufschlag	301	40	21,3			852,00	17040,00
2	Materialgemeinkosten					11,0%	151,00	3019,94
3	MATERIALKOSTEN						1523,70	30473,94
4	Fertigungskosten							
	Std	Ko-St.		Std_Satz				
4.1	56,00	F1		16,80		195,00%	2775,36	55507,20
5	Sondereinzelkosten der Fertigung						1000,00	20000,00
6	SUMME FERTIGUNGSKOSTEN						3775,36	75507,20
7	HERSTELLKOSTEN:						5299,06	105981,14
8	Verwaltungsgemeinkosten					10,50%	556,40	11128,02
9	Vertriebsgemeinkosten					19,50%	1033,32	20666,32
10	SELBSTKOSTEN:						6888,77	137775,48
11	Kalk. Gewinn %:					12,5%	861,10	17221,94
12	Provision					3,0%	239,69	4793,73
13	NETTO-PREIS:						7989,56	159791,15
14	Frachtkosten des Auftrags					100	5,00	100,00
15	Verpackungskosten des Auftrags					300	15,00	300,00
16	Skonto					2,0%	163,46	3269,21
17	ANGEBOTSPREIS (netto):						8173,02	163460,36
18	MwSt					19,0%	1552,87	31057,47
19	ANGEBOTSPREIS (brutto):						9725,89	194517,83

Abb. A5.7a Das Layout beim Ausgeben

A5.8) **LITERATUR**

- Investition und Finanzierung von Ulrich Ermschel et al., ISBN-978-3-7908-2745-3, Springer-Verlag Berlin Heidelberg 2011(https://kataloge.uni-hamburg.de/DB=2/SET=5/TTL=1/SHW?FRST=3)
- Automatisierung und Programmierung mit Excel 2010 (www.rrzn.uni-hannover.de/buecher.html)
- VBA – Programmierung mit Excel von Johannes Gogolok (VBA mit Excel von Uni-Hagen.pdf)

A6) Aufgabe: Eine Datenbank für die Aufzinsung der Kapitalanlagen

Beschreibung

Der Zins, als Kompensation für den zwischenzeitlichen Konsumverzicht des Kapitalgebers, stellt eine von vier Kategorien der Zinsrechnung dar. Er ergibt sich in der ersten Zinsperiode aus der Multiplikation des vereinbarten Zinssatzes i mit dem Anfangskapital K0. Weiterhin steht das Endkapital Kn im Interesse des Gläubigers bzw. Schuldners, welches wiederrum von der Laufzeit n des Finanzkontraktes abhängig ist. Der Zinssatz i wird hierbei auch als Nominalzins bezeichnet. Daraus lassen sich die Formeln für das Endkapital bei einfacher Verzinsung und bei Zinseszinsrechnung ermitteln.

Bei dieser Aufgabe sollen die Datensätze für die `Person`, die `Kapitalanlage`, das `Anfangskapital` und den `Zinssatz in Prozent` über eine Userform in der Stammdatentabelle verwaltet werden. Mithilfe der Formel für die Zinsrechungen (Abb. A2.5a) und der Angabe von der Laufzeit sollen die verfügbaren und berechneten Daten in die Ausgabetabelle übertragen werden.

Wichtige Funktionen

Aufbau einer Exceldatei zu einem automatisierten Zinsrechnung mit folgenden Funktionen:

- Umgang mit den Zellenformaten und Wertintervallen
- Berücksichtigung der wichtigen Aspekten eines Programms wie Benutzerfreundlichkeit, Korrektheit, Robustheit
- Programmierung und Umgang mit den Kontrollkätschen
- Konzeption und Realisation der Algorithmen zu der Aufgabestellung
- Umgang mit den Diagrammen

Vorgehensweise

Erstellen Sie eine Exceldatei mit dem Namen axxxxxxx.xLsm (xxxxxxx steht für Ihre Matrikelnummer). In dieser Aufgabe werden zwei Tabellen (`tblZinsEin` als Stammdatentabelle und `tblZinsAusneu` als Ausgabetabelle), eine Userform mit dem Namen `frmZinsneu` und ein Diagrammblatt zur grafischen Darstellung `DZinsAus` benötigt.
A6.1) Das Tabellenblatt (Tab. A6.1a) kann `tblZinsEin` heißen und beinhaltet Datensätze für die `Person` (Spalte A), die `Kapitalanlage` (Spalte B), das `Anfangskapital` (Spalte C) und den `Zinssatz` (Spalte D).

	A	B	C	D
1	Frau Meyer	Deutsche Anleihen	120.000,00 €	2%
2	Herr Bandegani	Kreditvergabe	60.000,00 €	6,50%
3	Herr Müler	Immobilien	150.000,00 €	3,20%
4	Herr Schmidt	Aktien	140.000,00 €	8%
5				
6	**Person**	**Kapitalanlage**	**Anfangskapital**	**Zinssatz**

Tab. A6.1a `tblZinsEin`

A6.2) Das Userformblatt (Abb. 6.2a) kann `frmZinsneu` heißen und besteht aus den folgenden Elementen:

Pos.	Bezeichnung (Caption)	Steuerelement	Name
1	Person:	Beschriftungsfeld	Nicht ändern
2	keine	Kombinationsfeld	cboPerson
3	Kapitalanlage:	Beschriftungsfeld	Nicht ändern
4	keine	Textfeld	txtAnlage
5	Anfangskapital:	Beschriftungsfeld	Nicht ändern
6	keine	Textfeld	txtKapital
7	Prozent	Beschriftungsfeld	Nicht ändern
8	keine	Textfeld	txtProzent
9	Zusatzdaten	Rahmen	Nicht ändern
10	Laufzeit im Jahr:	Beschriftungsfeld	Nicht ändern
11	keine	Kombinationsfeld	cboLaufzeit
12	Einfache Zinsrechnung:	Checkbox	chkEZins
13	Zinsenzinsrechnung:	Checkbox	chkZZins
14	Ablegen	Befehlsschaltfläche	cmdAblegen
15	Löschen	Befehlsschaltfläche	cmdloeschen
16	Ausgeben	Befehlsschaltfläche	cmdAusgeben
17	Beenden	Befehlsschaltfläche	cmdBeenden
18	Kennlinie…	Befehlsschaltfläche	cmdkennlinien
19	Formeln…	Befehlsschaltfläche	cmdInfo
20	keine	Drehfeld (SpinButton)	spnProzent

Erläuterung zu dem Programmablauf

A6.2.1) Beim Starten des Userformblatts (Abb. 6.2a) müssen alle Einträge der ersten Spalte von der Tabelle (Tab. A6.1a) in das Kombinationsfeld `cboPerson` eingefügt werden.

A6.2.2) Wenn ein Eintrag für die Person gewählt wird, muss der entsprechende Datensatz aus der Tabelle (Tab. A6.1a) in den Textfeldern erscheinen.

A6.2.3) Wenn der Benutzer einen neuen Eintrag in das Userformblatt einträgt und auf „Ablegen“ klickt, muss der neue Datensatz in die Tabelle (Tab. A6.1a) hinzugefügt und anschließend sortiert werden.

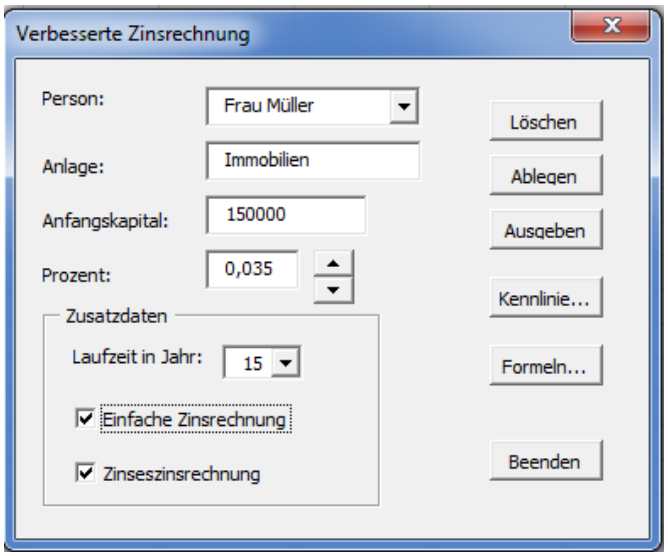


Abb. A6.2a `frmZinsneu`

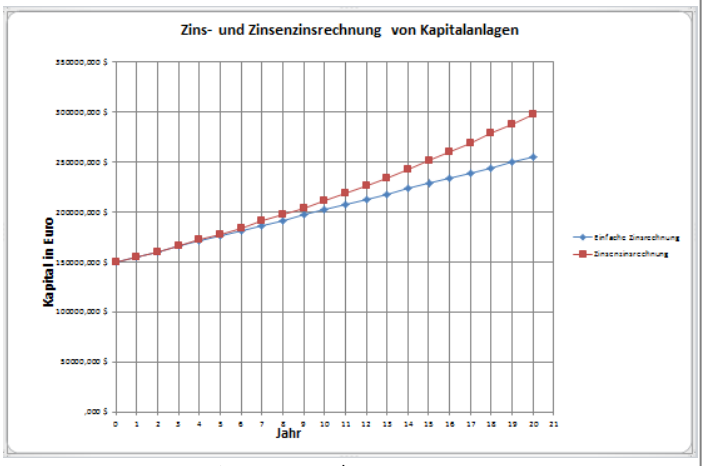
A6.2.4) Wenn der Benutzer einen Eintrag wählt, der schon in der Tabelle (Tab. A6.1a) existiert, muss der Datensatz beim Klicken auf „Ablegen“ ersetzt werden. (Hierbei soll eine

Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, dass der Datensatz ersetzt werden soll.)
A6.2.5) Wenn der Benutzer einen Eintrag wählt und auf die Befehlsschaltfläche „Löschen“ klickt, muss der Datensatz aus der Tabelle (Tab. A6.1a) gelöscht werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, die besagt, ob der Benutzer sicher ist, dass der Datensatz gelöscht werden soll.)
A6.2.6) Wenn der Benutzer auf die Befehlsschaltfläche „Ausgeben“ klickt, müssen alle in der Userform (Abb. A6.2a) augewälten Daten mit den berechneten Werten für den Kapitalwert (Abb. A6.5a) in die Ausgabetabelle (Tab. A6.3a) übertragen werden. (Hierbei soll eine Sicherheitsmeldung ausgegeben werden, wenn der Benutzer keinen Eintrag gewählt hat.)
A6.2.7) Die Befehlsschaltfläche „Beenden“ blendet die Userform aus.

A6.3) Erstellen Sie eine Tabelle (Tab. A6.3a) mit dem Namen `tblZinsAus`. Die Werte für die `Person` (Zeile 2/Spalte B), die `Kapitalanlage` (Zeile 3/Spalte B), das `Anfangskapital` (Zeile 4/Spalte B), den `Prozentsatz` (Zeile 5/Spalte B), die `Laufzeit im Jahr` (Zeile 6 /Spalte B) müssen von der Userform (Abb. A6.2a) in diese Tabelle übertragen werden. Die zu der Laufzeit im Jahr errechneten Kapitalwerte (Abb. A6.5a) müssen ab Zeile 9 in die Spalte B platziert werden.
A6.4) Erstellen Sie ein Diagrammblatt (Tab. A6.4a) und bennen Sie es mit dem Namen `DZinsAus` (Tab. A6.4a). Dabei beziehen die x-Achse auf das Jahr (Spalte A / Zeilen 9 bis 29) und die y-Achse auf den Kapitalwert (Spalte B / Zeilen 9 bis 29) von der Tabelle `tblZinsAusneu`(Tab. A6.3a).

	A	B	C
1	Aufzinsung für die Kapitalanlagen		
2	Investmentperson:	Frau Müller	
3	Kapitalanlage:	Immobilien	
4	Anfangskapital:	150.000,00 €	
5	Prozentsatz:	0,035	
6	Laufzeit im Jahr:	20	
7			
8	Jahr	Kapitalwert: Einfache Zinsrechnung	Kapitalwert: Zinsenzinsrechnung
9	0	150.000,00 €	150.000,00 €
10	1	155.250,00 €	155.250,00 €
11	2	160.500,00 €	160.683,75 €
12	3	165.750,00 €	166.307,68 €
13	4	171.000,00 €	172.128,45 €
14	5	176.250,00 €	178.152,95 €
15	6	181.500,00 €	184.388,30 €
16	7	186.750,00 €	190.841,89 €
17	8	192.000,00 €	197.521,36 €
18	9	197.250,00 €	204.434,60 €
19	10	202.500,00 €	211.589,81 €
20	11	207.750,00 €	218.995,46 €
21	12	213.000,00 €	226.660,30 €
22	13	218.250,00 €	234.593,41 €
23	14	223.500,00 €	242.804,18 €
24	15	228.750,00 €	251.302,32 €
25	16	234.000,00 €	260.097,91 €
26	17	239.250,00 €	269.201,33 €
27	18	244.500,00 €	278.623,38 €
28	19	249.750,00 €	288.375,20 €
29	20	255.000,00 €	298.468,33 €

Tab. A6.3a `tblZinsAusneu`



Tab. A6.4a `DZinsAusneu`

Abb. A6.5a **`Formeln zur Verzinsung`**

1) Einfache Verzinsung: $K_n = K_0(1 + i \,.\, n)$

2) Zinseszinsrechnung: $K_n = K_0(1 + i)^n$

Programmablaufplan

Der Programmablaufplan (Abb. A6.6a) bezieht sich auf die Abschnitte A6.2.1 – A6.2.7 Entwickeln und realisieren Sie bitte einen Programmablaufplan, in dem kurze Codes und vereinfachte Aktionen für den Benutzer ermöglicht werden.

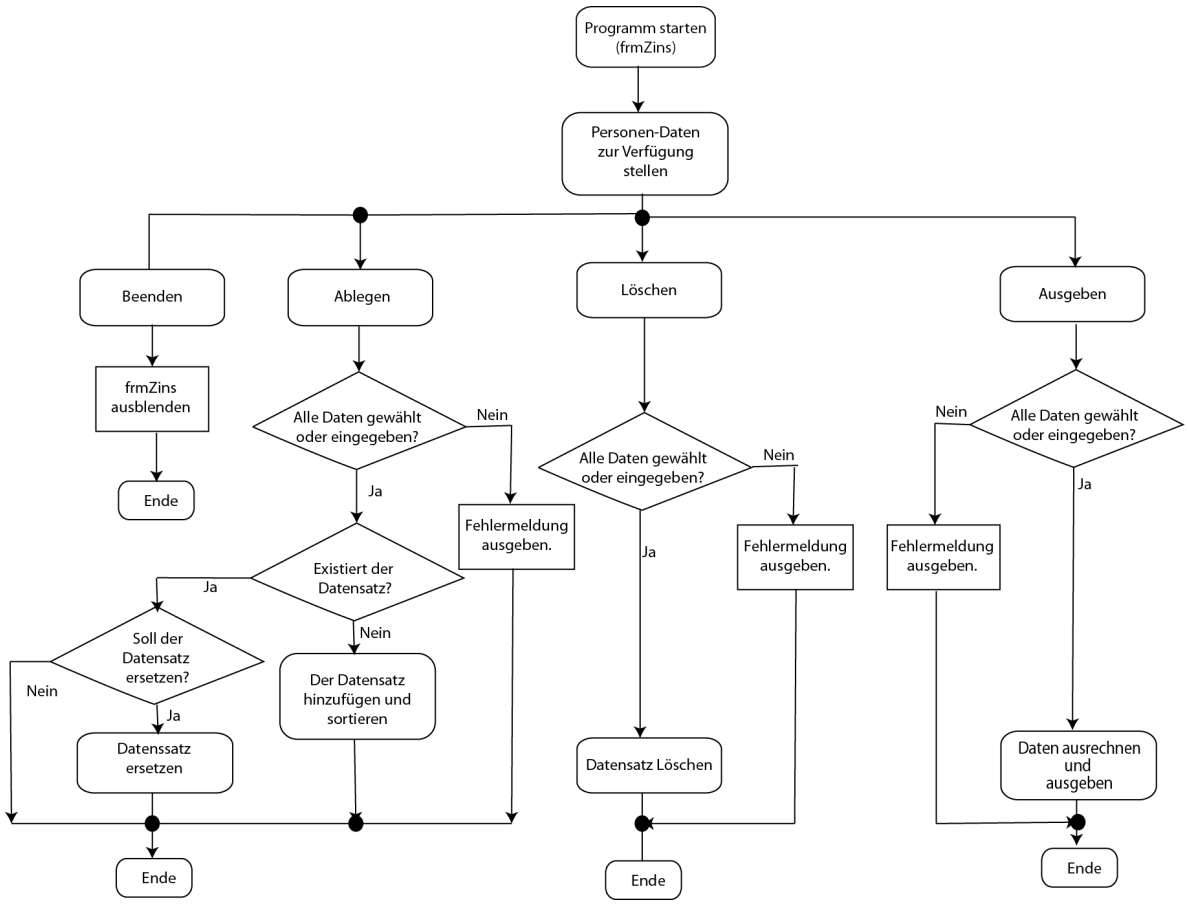


Abb. A6.6a Programmablaufplan zu der Verzinsung

A2.6) Folgende VBA-Codes dienen dazu, in den Abschnitten A6.2.1-A6.2.7 beschriebenen Programmablaufplan zu implementieren:

```
Dim varWelcherEintrag As Integer
Dim varCurrent As Byte
Dim varFeldMatrix(1 To 4) As Variant
Dim varZinsMatrix(0 To 1) As Byte
```

```
Private Sub userform_Initialize()
   Me.chkEZins.Value = False
   Me.chkZZins.Value = False
   Maskeloeschen
   Einfuegen "tblZinsEin", Me.cboPerson, 1
   Me.cboLaufzeit.AddItem 5
   Me.cboLaufzeit.AddItem 8
   Me.cboLaufzeit.AddItem 10
   Me.cboLaufzeit.AddItem 12
   Me.cboLaufzeit.AddItem 15
   Me.cboLaufzeit.AddItem 20
   Worksheets("tblZinsEin").Activate
End Sub
```

```
Sub Einfuegen(Tabellenname As String, KomboBox As Object, Spaltennummer As Integer)
   Dim varEndwert As Integer
   Dim i As Integer
```

```
    varEndwert = Worksheets(Tabellenname).Cells(1, 1).CurrentRegion.Rows.Count
    i = 0
    For i = 1 To varEndwert
        KomboBox.AddItem Worksheets(Tabellenname).Cells(i, Spaltennummer).Value
    Next i
End Sub
Private Sub cboPerson_Click()
    varWelcherEintrag = Me.cboPerson.ListIndex + 1
    Me.txtAnlage.Text = Worksheets("tblZinsEin").Cells(varWelcherEintrag, 2).Value
    Me.txtKapital.Text = Worksheets("tblZinsEin").Cells(varWelcherEintrag, 3).Value
    Me.txtProzent.Text = Worksheets("tblZinsEin").Cells(varWelcherEintrag, 4).Value
End Sub
Private Sub cboPerson_Enter()
    Me.txtAnlage.Text = ""
    Me.txtKapital.Text = ""
    Me.txtProzent.Text = ""
    Me.cboLaufzeit.Text = ""
    Me.chkEZins.Value = False
    Me.chkZZins.Value = False
End Sub
Private Sub cboPerson_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Me.txtAnlage.Text = ""
    Me.txtKapital.Text = ""
    Me.txtProzent.Text = ""
    Me.cboLaufzeit.Text = ""
    Me.chkEZins.Value = False
    Me.chkZZins.Value = False
End Sub
Private Sub cmdloeschen_Click()
    If Felderpruefen Then
        Meldung = "Wollen Sie wirklich den Datensatz löschen?"
        Titel = "Datensatz löschen"
        Stil = vbYesNo + vbCritical + vbDefaultButton2
        Ergebnis = MsgBox(Meldung, Stil, Titel)
        If Ergebnis = vbYes Then
            Worksheets("tblZinsEin").Rows(varWelcherEintrag).Delete Shift:=x1up
            Maskeloeschen
            Einfuegen "tblZinsEin", Me.cboPerson, 1
        Else
            Exit Sub
        End If
    End If
End Sub
Sub Maskeloeschen()
    Me.cboPerson.Clear
    Me.cboPerson.Text = ""
    Me.txtAnlage.Text = ""
    Me.txtKapital.Text = ""
    Me.txtProzent.Text = ""
    Me.cboLaufzeit.Text = ""
End Sub
Private Sub cmdAblegen_Click()
    If Felderpruefen Then
        If Me.cboPerson.Text > "" And varWelcherEintrag = 0 Then
            Neueintragen
            Maskeloeschen
            Einfuegen "tblZinsEin", Me.cboPerson, 1
            Worksheets("tblZinsEin").Activate
        End If
        If Me.cboPerson.Text > "" And varWelcherEintrag > 0 Then
            If Worksheets("tblZinsEin").Cells(varWelcherEintrag, 1).Value = Me.cboPerson.Text Then
```

```
            Dim Meldung As String
            Dim Titel As String
            Dim Stil As Integer
            Dim Ergebnis As Integer
            Stil = vbYesNo + vbCritical + vbDefaultButton2
            Meldung = "Wollen Sie den Datensatz ersetzen?"
            Titel = "Datensatz ersetzen"
            Ergebnis = MsgBox(Meldung, Stil, Titel)
            If Ergebnis = vbYes Then
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 2).NumberFormat = "@"
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 2).Value = Me.txtAnlage.Text
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 3).NumberFormat = "#,##0.00 $"
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 3).Value = Me.txtKapital.Text
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 4).NumberFormat = "0.00%"
               Worksheets("tblZinsEin").Cells(varWelcherEintrag, 4).Value = Me.txtProzent.Text
               Maskeloeschen
               Einfuegen "tblZinsEin", Me.cboPerson, 1
               Worksheets("tblZinsEin").Activate
            Else
               Exit Sub
            End If
         Else
            Neueintragen
            Maskeloeschen
            Einfuegen "tblZinsEin", Me.cboPerson, 1
            Worksheets("tblZinsEin").Activate
         End If
      End If
   End If
End Sub
```

```
Sub Neueintragen()
   Dim varNeueZeile As Integer
   varNeueZeile = Worksheets("tblZinsEin").Cells(1, 1).CurrentRegion.Rows.Count + 1
   Worksheets("tblZinsEin").Cells(varNeueZeile, 1).NumberFormat = "@"
   Worksheets("tblZinsEin").Cells(varNeueZeile, 1).Value = Me.cboPerson.Text
   Worksheets("tblZinsEin").Cells(varNeueZeile, 2).NumberFormat = "@"
   Worksheets("tblZinsEin").Cells(varNeueZeile, 2).Value = Me.txtAnlage.Text
   Worksheets("tblZinsEin").Cells(varNeueZeile, 3).NumberFormat = "#,##0.00 $"
   Worksheets("tblZinsEin").Cells(varNeueZeile, 3).Value = Me.txtKapital.Text
   Worksheets("tblZinsEin").Cells(varNeueZeile, 4).NumberFormat = "0.00%"
   Worksheets("tblZinsEin").Cells(varNeueZeile, 4).Value = Me.txtProzent.Text
   With Worksheets("tblZinsEin")
      .Range(.Cells(1, 1), .Cells(varNeueZeile, 4)).Sort Key1:= _
      Worksheets("tblZinsEin").Columns("A")
   End With
   varWelcherEintrag = varNeueZeile
End Sub
```

```
Private Sub cmdAusgeben_Click()
   If Felderpruefen = True And Me.cboLaufzeit.Text > "" Then
      Worksheets("tblZinsAusneu").Activate
      Ausgabeloeschen
      Worksheets("tblZinsAusneu").Cells(2, 1).NumberFormat = "@"
      Worksheets("tblZinsAusneu").Cells(2, 1).Value = "Investmentperson:"
      Worksheets("tblZinsAusneu").Cells(2, 2).NumberFormat = "@"
```

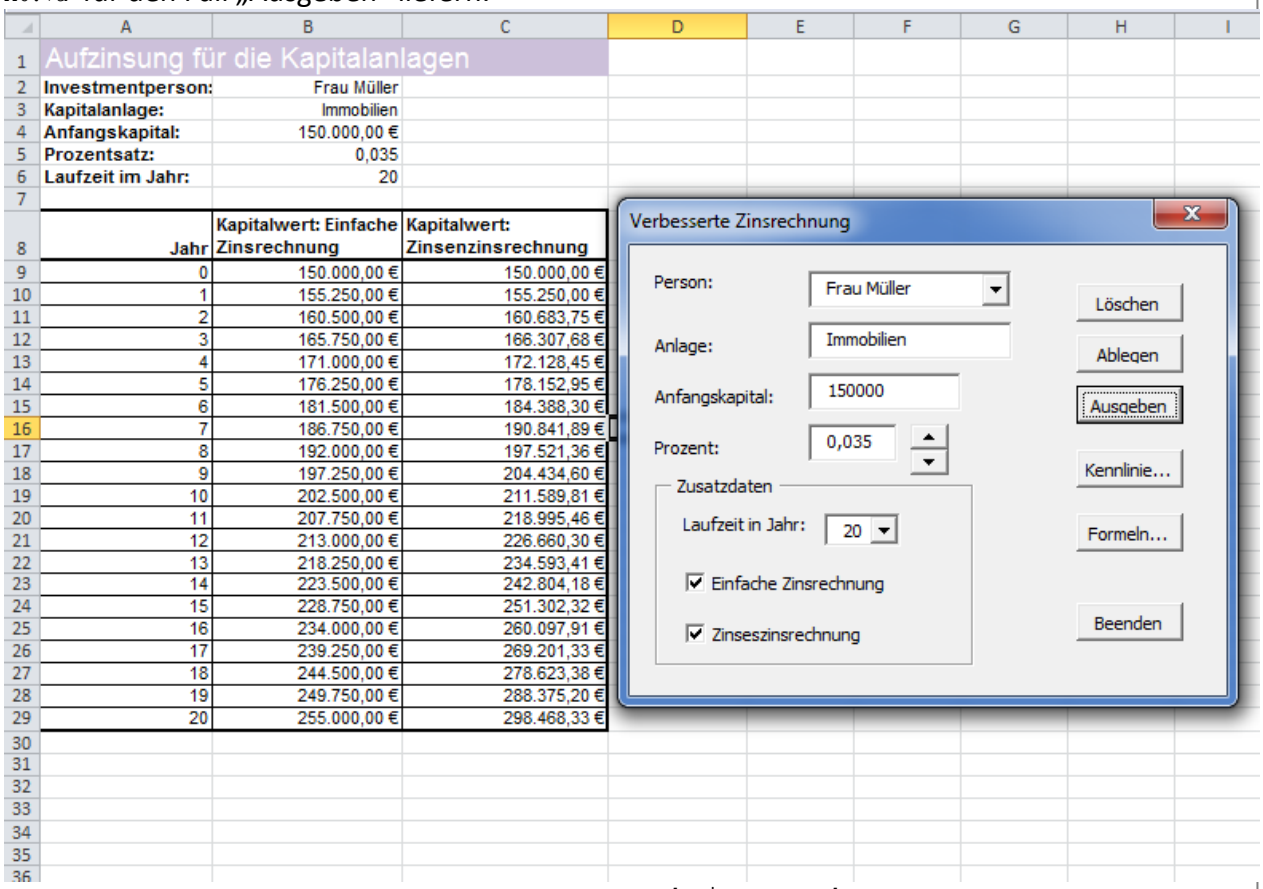
```
Worksheets("tblZinsAusneu").Cells(2, 2).Value = Me.cboPerson.Text
Worksheets("tblZinsAusneu").Cells(3, 1).NumberFormat = "@"
Worksheets("tblZinsAusneu").Cells(3, 1).Value = "Kapitalanlage:"
Worksheets("tblZinsAusneu").Cells(3, 2).NumberFormat = "@"
Worksheets("tblZinsAusneu").Cells(3, 2).Value = Me.txtAnlage.Text
Worksheets("tblZinsAusneu").Cells(4, 1).NumberFormat = "@"
Worksheets("tblZinsAusneu").Cells(4, 1).Value = "Anfangskapital:"
Worksheets("tblZinsAusneu").Cells(4, 2).NumberFormat = "#,##0.00 $"
Worksheets("tblZinsAusneu").Cells(4, 2).Value = Me.txtKapital.Text
Worksheets("tblZinsAusneu").Cells(5, 1).NumberFormat = "@"
Worksheets("tblZinsAusneu").Cells(5, 1).Value = "Prozentsatz:"
Worksheets("tblZinsAusneu").Cells(5, 2).NumberFormat = "0.00%"
Worksheets("tblZinsAusneu").Cells(5, 2).Value = Me.txtProzent.Text
Worksheets("tblZinsAusneu").Cells(6, 2).NumberFormat = "0"
Worksheets("tblZinsAusneu").Cells(6, 2).Value = Me.cboLaufzeit.Text
Worksheets("tblZinsAusneu").Cells(8, 1).NumberFormat = "@"
Worksheets("tblZinsAusneu").Cells(8, 1).Value = "Jahr"
Worksheets("tblZinsAusneu").Cells(8, 2).NumberFormat = "@"
Worksheets("tblZinsAusneu").Cells(8, 2).Value = "Kapitalwert: Einfache Zinsrechnung"
Worksheets("tblZinsAusneu").Cells(8, 3).NumberFormat = "@"
Worksheets("tblZinsAusneu").Cells(8, 3).Value = "Kapitalwert: Zinsenzinsrechnung"
Select Case ZustandErmitteln
  Rem 1. Zustand "Keine Zinsart gewählt"
  Case 0
    MsgBox "Sie haben keine Zinsart gewählt!"
    Exit Sub
  Rem 2. Zustand "Einfache Zinsrechnung gewählt"
  Case 1
    i = 0
    For i = 0 To Me.cboLaufzeit.Text
      Worksheets("tblZinsAusneu").Cells(9 + i, 1).Value = i
      Worksheets("tblZinsAusneu").Cells(9 + i, 2).NumberFormat = "#,##0.00 $"
      Worksheets("tblZinsAusneu").Cells(9 + i, 2).Value = Me.txtKapital.Text * (1 + Me.txtProzent.Text * i)
    Next i
  Rem 3. Zustand "Zinsenzinsrechnung gewählt"
  Case 2
    i = 0
    For i = 0 To Me.cboLaufzeit.Text
      Worksheets("tblZinsAusneu").Cells(9 + i, 1).Value = i
      Worksheets("tblZinsAusneu").Cells(9 + i, 3).NumberFormat = "#,##0.00 $"
      Worksheets("tblZinsAusneu").Cells(9 + i, 3).Value = Me.txtKapital.Text * (1 + Me.txtProzent.Text * 1) ^ i
    Next i
  Rem 4. Zustand "Einfache Zins- und Zinsenzinsrechnung gewählt"
  Case 3
    i = 0
    For i = 0 To Me.cboLaufzeit.Text
      Worksheets("tblZinsAusneu").Cells(9 + i, 1).Value = i
      Worksheets("tblZinsAusneu").Cells(9 + i, 2).NumberFormat = "#,##0.00 $"
      Worksheets("tblZinsAusneu").Cells(9 + i, 2).Value = Me.txtKapital.Text * (1 + Me.txtProzent.Text * i)
    Next i
    i = 0
    For i = 0 To Me.cboLaufzeit.Text
      Worksheets("tblZinsAusneu").Cells(9 + i, 1).Value = i
      Worksheets("tblZinsAusneu").Cells(9 + i, 3).NumberFormat = "#,##0.00 $"
```

```
            Worksheets("tblZinsAusneu").Cells(9 + i, 3).Value = Me.txtKapital.Text * (1 + Me.txtProzent.Text * 1) ^ i
        Next i
    End Select
  Else
    MsgBox "Laufzeit ist nicht gewählt worden!"
  End If
End Sub
Sub Ausgabeloeschen()
  For i = 0 To 4
    Worksheets("tblZinsAusneu").Cells(2 + i, 2).Value = Null
  Next i
  For i = 0 To 20
    Worksheets("tblZinsAusneu").Cells(i + 9, 1).Value = Null
    Worksheets("tblZinsAusneu").Cells(i + 9, 2).Value = Null
    Worksheets("tblZinsAusneu").Cells(i + 9, 3).Value = Null
  Next i
End Sub
Function Felderpruefen() As Boolean
  Dim varFeldMatrix(1 To 4) As Variant
  Dim i As Integer
  i = 0
  For i = 1 To 4
    varFeldMatrix(i) = 0
  Next i
  varFeldMatrix(1) = Me.cboPerson.Text
  varFeldMatrix(2) = Me.txtAnlage.Text
  varFeldMatrix(3) = Me.txtKapital.Text
  varFeldMatrix(4) = Me.txtProzent.Text
  If varFeldMatrix(1) > "" Then
    Felderpruefen = True
  Else
    MsgBox "Sie müssen in dem Kombinationsfeld eine Person eingeben oder wählen!"
    Felderpruefen = False
    Me.cboPerson.BackColor = RGB(127, 128, 0)
    Exit Function
  End If
  If varFeldMatrix(2) > "" Then
    Felderpruefen = True
  Else
    MsgBox "Sie müssen in dem Textfeld eine Kapitalanlage eingeben!"
    Felderpruefen = False
    Me.txtAnlage.BackColor = RGB(127, 128, 120)
    Exit Function
  End If
  If varFeldMatrix(3) > "" Then
    Felderpruefen = True
  Else
    MsgBox "Sie müssen in dem Textfeld einen Kapitalwert eingeben!"
    Felderpruefen = False
    Me.txtKapital.BackColor = RGB(127, 128, 50)
    Exit Function
  End If
  If varFeldMatrix(4) > "" Then
    Felderpruefen = True
  Else
    MsgBox "Sie müssen in dem Textfeld ein Kommazahl für Prozent eingeben!"
    Felderpruefen = False
    Me.txtProzent.BackColor = RGB(127, 128, 50)
    Exit Function
```

```
    End If
End Function
Function ZustandErmitteln() As Byte
    Dim varWert As Byte
    varWert = 0
    For i = 0 To 1
        varWert = varWert + (2 ^ i) * varZinsMatrix(i)
    Next i
    ZustandErmitteln = varWert
End Function
Private Sub chkEZins_Click()
    If Me.chkEZins.Value = True Then
        varZinsMatrix(0) = 1
    Else
        Me.chkEZins.Value = False
        varZinsMatrix(0) = 0
    End If
End Sub
Private Sub chkZZins_Click()
    If Me.chkZZins.Value = True Then
        varZinsMatrix(1) = 1
    Else
        Me.chkZZins.Value = False
        varZinsMatrix(1) = 0
    End If
End Sub
Private Sub spnProzent_change()
    Dim varValue As Byte
    varValue = Me.spnProzent.Value
    If Me.txtProzent.Value > "" Then
        If varValue > varCurrent And varValue <= 100 Then
            Me.txtProzent.Value = Me.txtProzent.Value + 0.005
        ElseIf varValue < varCurrent And varValue >= 0 Then
            Me.txtProzent.Value = Me.txtProzent.Value - 0.005
        Else
            Me.txtProzent.Value = 0.005
        End If
        varCurrent = varValue
    Else
        Me.txtProzent.Value = Me.txtProzent.Value
    End If
End Sub
Private Sub cmdkennlinien_Click()
    Sheets("DZinsAusneu").Select
    ActiveChart.ChartArea.Select
    ActiveChart.ChartTitle.Select
    Select Case ZustandErmitteln
        Rem 1. Zustand "Keine Zinsart gewählt"
        Case 0
            ActiveChart.ChartTitle.Text = "keine Zinsberechnung"
        Rem 2. Zustand "Einfache Zinsrechnung gewählt"
        Case 1
            ActiveChart.ChartTitle.Text = "Einfache Zinsrechnung von Kapitalanlagen"
        Rem 3. Zustand "Zinsenzinsrechnung gewählt"
        Case 2
          ActiveChart.ChartTitle.Text = "Zinsenzinsrechnung von Kapitalanlagen"
        Rem 4. Zustand "Einfache Zins- und Zinsenzinsrechnung gewählt"
        Case 3
            ActiveChart.ChartTitle.Text = "Zins- und Zinsenzinsrechnung von Kapitalanlagen"
        End Select
End Sub
```

```
Private Sub cmdInfo_Click()
   Worksheets("tblZinsInfo").Activate
End Sub
Private Sub cmdBeenden_Click()
   Me.Hide
End Sub
```

A6.7) Wie schon im Abschnitt A6.2.6 erläutert wurde, muss das Programm als Ergebnis Abb. A6.7a für den Fall „Ausgeben“ liefern.



	A	B	C
1	Aufzinsung für die Kapitalanlagen		
2	Investmentperson:	Frau Müller	
3	Kapitalanlage:	Immobilien	
4	Anfangskapital:	150.000,00 €	
5	Prozentsatz:	0,035	
6	Laufzeit im Jahr:	20	
7			
8	Jahr	Kapitalwert: Einfache Zinsrechnung	Kapitalwert: Zinsenzinsrechnung
9	0	150.000,00 €	150.000,00 €
10	1	155.250,00 €	155.250,00 €
11	2	160.500,00 €	160.683,75 €
12	3	165.750,00 €	166.307,68 €
13	4	171.000,00 €	172.128,45 €
14	5	176.250,00 €	178.152,95 €
15	6	181.500,00 €	184.388,30 €
16	7	186.750,00 €	190.841,89 €
17	8	192.000,00 €	197.521,36 €
18	9	197.250,00 €	204.434,60 €
19	10	202.500,00 €	211.589,81 €
20	11	207.750,00 €	218.995,46 €
21	12	213.000,00 €	226.660,30 €
22	13	218.250,00 €	234.593,41 €
23	14	223.500,00 €	242.804,18 €
24	15	228.750,00 €	251.302,32 €
25	16	234.000,00 €	260.097,91 €
26	17	239.250,00 €	269.201,33 €
27	18	244.500,00 €	278.623,38 €
28	19	249.750,00 €	288.375,20 €
29	20	255.000,00 €	298.468,33 €

Abb. A6.7a Das Layout beim Ausgeben

A6.8) **LITERATUR**

- Investition und Finanzierung von Ulrich Ermschel et al., ISBN-978-3-7908-2745-3, Springer-Verlag Berlin Heidelberg 2011(https://kataloge.uni-hamburg.de/DB=2/SET=5/TTL=1/SHW?FRST=3)
- Automatisierung und Programmierung mit Excel 2010 (www.rrzn.uni-hannover.de/buecher.html)
- VBA – Programmierung mit Excel von Johannes Gogolok (VBA mit Excel von Uni-Hagen.pdf)