Schwerpunkte ebener Flächen

VBA - Visual Basic für Excel im Unterricht

Erwin Podenstorfer DSK-500: EXCEL*.XLS

Ab Excel 5.0 ist dieses Tabellenkalkulationsprogramm mit einer neuen Makrosprache (Visual Basic) mit allen Merkmalen einer „echten" Programmiersprache ausgestattet. Durch den objektorientierten Ansatz kann die Sprache beliebig erweitert werden. Alle Programme des Paketes Microsoft-Office sollen mit VBA ausgestattet werden, Grund genug, die neuen Möglichkeiten von Excel im Informatikunterricht den Schülern eines 3. Jahrganges im Rahmen eines kleinen Projektes nahezubringen.

Wozu Makroprogrammierung?

Excel bietet in seiner „normalen" Anwendung eine fast unüberschaubare Anzahl von Möglichkeiten. Wozu braucht man überhaupt Makros? Warum sollen wir uns an eine vollkommen neue Programmiersprache gewöhnen?

- Mit VBA lassen sich eigene Tabellenfunktionen programmieren, die einfacher anzuwenden sind als manche excelinterne Funktion. Man denke etwa an die `WENN()` Funktion, die, wenn sie in geschachtelter Form verwendet werden muß, praktisch unlesbar ist und eine Zumutung ist.
- Man kann Excel nach eigenen Vorstellungen konfigurieren und auf einfache Weise eine effizientere Bedienung erreichen.
- Man kann komplexe Arbeitsschritte - etwa das Ausfüllen von Formularen - durch sogenannte „intelligente" Formulare strukturieren und erleichtern.
- Man kann immer wieder auftretende Arbeitsvorgänge automatisieren. Das empfiehlt sich besonders dann, wenn regelmäßig große Datenmengen anfallen, die analysiert, verarbeitet und grafisch aufbereitet werden sollen.
- Man kann eigenständige Excel-Programme erstellen, die sich durch eigene Menüs, eigene Dialoge usw. auszeichnen. Damit lassen sich Excel-Anwendungen in ihrer Bedienung soweit vereinfachen, daß sie von anderen Personen (auch von Excel-Laien) ohne lange Einschulung bedient werden können.

Vorbereitungen für das Projekt

Beim Arbeiten mit dem Betriebssystem Windows wurden die Schüler besonders auf die objekt- und ereignisorientierte Arbeitsweise von Windows Programmen hingewiesen.

Beim Programmieren mit Visual Basic wurden eigenständig einfache Windowsprogramme erzeugt. Die Begriffe Objekt, Eigenschaft, Methode, Ereignis, Ereignisprozedur wurden dabei weiter gefestigt.

Schließlich erfolgte eine Einführung in das Arbeiten mit Excel 5.0.

Problemstellung

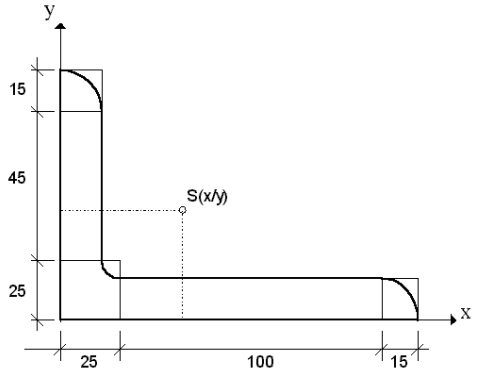

Abbildung 1

Von ebenen Flächenstücken, die aus bestimmten Teilfiguren additiv und subtraktiv zusammengesetzt sind, sollen die Koordinaten des Flächenschwerpunktes bestimmt werden.

Zulässige Teilfiguren sind

- Quadrat
- Rechteck
- Rechtwinkeliges Dreieck (Katheten parallel zu den Koordinatenachsen)
- Allgemeines Dreieck
- Kreis
- Ellipse
- Halbkreis (Halbellipse), Durchmesser parallel zur x-Achse
- Halbkreis (Halbellipse), Durchmesser parallel zur y-Achse
- Viertelkreis (Viertelellipse), Begrenzungsradien parallel zu den Koordinatenachsen

Analyse

Die Problemanalyse erfolgte im Unterrichtsgegenstand Statik.

Konzept

Die Erledigung der Aufgabe soll auf mehrere Tabellenblätter verteilt werden. Ein Tabellenblatt (Name `ERGEBNIS`) soll als Summenblatt ausgelegt werden und als Ergebnis die Koordinaten x und y des Gesamtschwerpunktes beinhalten.

Für jeden Teilflächenfall ist ein eigenes Tabellenblatt vorzusehen. Zur leichteren Orientierung bei der Dateneingabe ist eine entsprechende Grafik zu erstellen. Die ***Abbildungen 2*** und ***3*** zeigen das fertige Summenblatt und das Tabellenblatt für den Teilflächenfall Rechteck.

Durchführung

Die Aufgabe wurde zuerst mit den herkömmlichen Möglichkeiten von Excel (ohne VBA) bearbeitet. Einige Aufgaben wurden anschließend unter Verwendung von VBA gelöst. Nachdem gemeinsam die Bearbeitung eines Teilflächenfalles gemacht wurde, mußten die übrigen Fälle bzw. zusätzliche Fälle in Gruppenarbeit erledigt werden.

Aufgabe 1

Aktivieren des Tabellenblattes für eine bestimmte Teilfläche mit Hilfe einer Befehlsschaltfläche. Z.B. soll das Tabellenblatt `RECHTECK` durch Klicken der Befehlsschaltfläche mit der Aufschrift Rechteck aktiviert werden. Die erste Zelle der nächsten freien Zeile des Eingabebereiches des Tabellenblattes `RECHTECK` soll automatisch aktualisiert werden. Die Zeile soll automatisch numeriert werden.

Aufgabe 2

Installieren einer Befehlsschaltfläche „Drucken" für das Drucken des gerade aktiven Tabellenblattes.

Aufgabe 3

Installieren eines Drop-Down-Listenobjektes im Summenblatt, mit dem das zu druckende Tabellenblatt ausgewählt bzw. gedruckt werden kann.

Aufgabe 4

Nach dem Öffnen der Anwendung soll nur mehr das Hauptmenü von Excel und die Statuszeile sichtbar sein. Nach dem Schließen der Anwendung sollen alle vor dem Öffnen der Anwendung aktiv gewesenen Bedienungselemente wieder sichtbar sein.

Aufgabe 5

In jeder Teilflächentabelle soll nach jeder Eingabe im zulässigen Eingabebereich eine automatische Datenprüfung erfolgen. Nur Zahlen sind erlaubt.

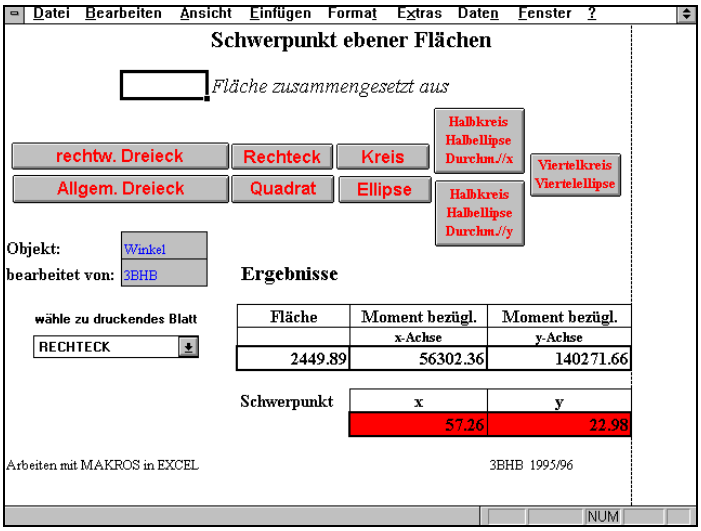

Abbildung 2

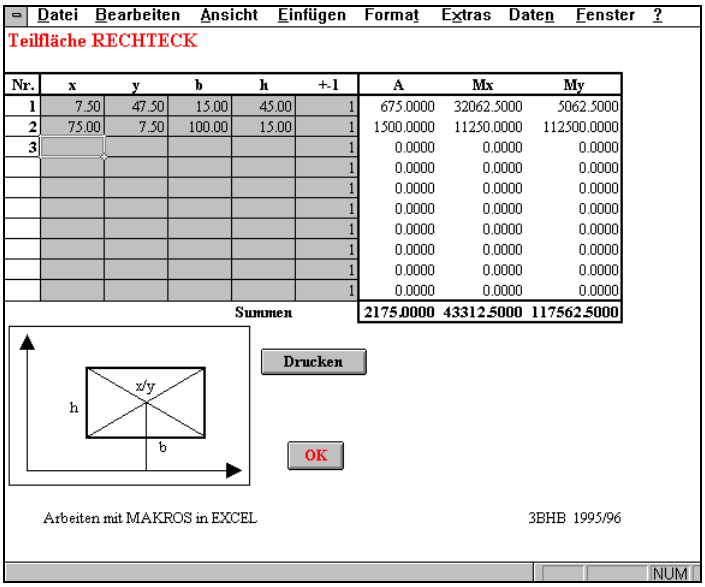

Abbildung 3

Die Steuerelemente in Excel

Excel 5.0 bietet 12 verschiedene Steuerelemente. Der Zugriff erfolgt am besten über die Symbolleiste DIALOG, die mit dem Menüpunkt *Ansicht-Symbolleisten-Dialog* aktiviert wird.

Abbildung 4

Die Steuerelemente sind der Reihe nach:

Bezeichnungsfeld	Darstellung von Text
Bearbeitungsfeld	Eingabe von Text, editierbar
Gruppenfeld	Zusammenfassung von Steuerelementen
Befehlsschaltfläche	
Kontrollkästchen	Einstellungen(wahr,falsch)
Optionsfeld	Auswahl
Listenfeld	Auswahl aus einer einfachen Liste
Drop-Down Listenfeld	
Kombinationsfeld (Liste-Text)	
Kombinationsfeld (Drop-Down-Text)	
Bildlaufleiste	Einstellung eines Wertes aus einem Zahlenbereich mit der Maus
Drehfeld	wie oben, ohne Schieber

Die 4 unteren Schaltflächen bedeuten der Reihe nach:

Steuerelement-Eigenschaften	Zugriff auf Eigenschaften
Code bearbeiten	Wechsel zum zugewiesenen Makro
Raster (ein/aus)	
Dialog ausführen	

Verwenden von Steuerelementen

Um einen Vorgang (Ereignisprozedur) durch Klicken eines Steuerelementes auszulösen, müssen 3 Schritte absolviert werden.

1. Schritt: Erzeugen des grafischen Symbols für das gewünschte Steuerelement. Dies geschieht, indem man im Dialogmenü (Figur 4) das gewünschte Symbol anklickt und auf dem Tabellenblatt ein Rechteck zeichnet. Die voreingestellte Bezeichnung kann dabei direkt im Symbol editiert werden. Größe und Position können beliebig verändert werden.

2. Schritt: Erstellen eines Makros oder einer VB-Prozedur, die bei Mausklick auf dem Steuerelement ablaufen soll. Ein Makro ist eine von Excel aufgezeichnete Folge von Benützeraktionen. Die Aufzeichnung erfolgt als VBA-Programm in einem automatisch der Anwendung hinzugefügten Modulblatt. Zwischen einer selbst geschriebenen Prozedur und einem vom System aufgezeichneten Makro besteht also kein Unterschied.

3. Schritt: Herstellen einer Verbindung zwischen dem Steuerelement und der im Schritt 2 erstellten Ereignisprozedur. (Für VB-Programmierer: damit ist das Click-Ereignis eines Steuerelementes geboren!)

Lösung der Aufgabe 2

Zunächst wird im Tabellenblatt RECHTECK nach Schritt 1 eine Befehlsschaltfläche gezeichnet und die Bezeichnung auf „Drucken“ geändert. Das Eingabefenster „Zuweisen“ müssen wir abbrechen, da die gewünschte Ereignisprozedur (Makro) noch nicht existiert. Die zugeordnete Ereignisprozedur lassen wir uns mit dem Menüpunkt *-Extras-Makro aufzeichnen-* aufzeichnen.. eben aufzeichnen. Als Makroname vergeben wir z.B. *akttabdrucken*. Die Befehlsfolge Menüpunkt *Datei-Drucken* und schließlich Menüpunkt *Extras-Makro aufzeichnen* -aufzeichnen beenden zum Beenden des Aufzeichnungsprozesses erzeugt in einem Modulblatt (Modul1) den Code

```
Option Explicit

'
' akttabdrucken Makro
' Makro am 1996-02-23 von EDV
' aufgezeichnet
'
'
Sub akttabdrucken()
  ActiveWindow.SelectedSheets.PrintOut Copies:=1
End Sub
```

Interessant sind bloß die letzten drei Zeilen. `ActiveWindow` ist ein Objekt und bezeichnet das aktive Fenster. `SelectedSheets` bezeichnet ein untergeordnetes Objekt und zwar das aktuelle Blatt und `Printout` ist die Methode, die den Ausdruck steuert. `Copies := 1` ist ein sogenannter benannter Parameter.

Nun müssen wir noch Schritt 3 vollziehen, nämlich der Schaltfläche *Drucken* die Prozedur Sub `akttabdrucken()` zuweisen. Dies erreichen wir, indem wir mit der rechten Maustaste auf die Schaltfläche *Drucken* klicken und im Kontextmenü den Punkt *Zuweisen...* wählen. In der erscheinenden Liste aller in der Anwendung vorhandenen Makros wählen wir den Eintrag `akttabdrucken`!

Wenn wir in der Statuszeile während des Druckvorganges die Mitteilung *„Das aktuelle Tabellenblatt wird gerade gedruckt“* sehen wollen, müssen wir in der Prozedur `akttabdrucken()` einige Zeilen Code einfügen.

```
Sub akttabdrucken()
  Application.DisplayStatusbar = True
  Application.Statusbar = "Das aktuelle Tabellenblatt
wird gerade gedruckt"
  ActiveWindow.SelectedSheets.PrintOut Copies:=1
  Application.Statusbar = ""
End Sub
```

`Application` ist der Name des umfassenden Objektes und bezeichnet die aktuelle Anwendung. `DisplayStatusbar` ist der Name einer Eigenschaft, die die Werte `True` oder `False` haben kann. `Statusbar` bezeichnet die Eigenschaft, die die anzuzeigende Meldung als Zeichenkette beinhaltet.

Lösung der Aufgabe 1

Diesmal schreiben wir die der Schaltfläche *„Rechteck“* im Summenblatt zuzuweisende Prozedur selbst. Es soll ja nicht nur das Tabellenblatt *„Rechteck“* aktiviert werden, sondern auch automatisch die erste leere Zelle im Eingabebereich (Spalte B), sodaß sofort mit der Eingabe begonnen werden kann. Gleichzeitig soll in der Spalte A die Numerierung um 1 erhöht werden!

```
Global tabnr as Integer

Sub Rechteck()
  tabnr = 3 ' wird später gebraucht
  tabellevorbereiten("Rechteck")
End Sub

Sub tabellevorbereiten(nr)
  Dim z As Integer
  Sheets(nr).Select
  z = eingabezeile(3; "B")
  'In Zelle(A,z)den Wert z-3 speichern
  Cells(z; "A").Value = z-3
  'Die Zelle schließlich aktivieren
  Cells(z; "B").Select
End Sub

Function eingabezeile(vzeile; s) As Integer
  'sucht in der ausgewählten Tabelle
  'in Spalte s ab der Zeile vzeile
  'die Position der ersten leeren
  'Zelle
  Dim i As Integer
  For i = vzeile To 13
    If IsEmpty(Cells(i; s).Value) Then Exit For
  Next i
  eingabezeile = i
End Function
```

Erklärungsbedürftig ist eigentlich nur die Anweisung

```
Cells(z; "A").Value = z-3
```

`Cells` ist eine als Funktion agierende Methode, die auf grund der Parameter ein Objekt erzeugt, nämlich die gewünschte Zelle. In der Eigenschaft `value` des erzeugten Objektes ist der Wert der Zelle gespeichert, der also mit obiger Anweisung verändert wird. Die Methode `Cells` kann nur auf ein aktuelles Tabellenblatt ausgeübt werden. Dieses wird vorher mit der Anweisung `Sheets(nr).Select` festgelegt. Die VBA-Funktion `IsEmpty()` prüft, ob eine Zelle leer ist.

Für die Erledigung von Aufgaben können beliebig VB-Prozeduren oder Funktionen geschrieben werden. Sie müssen sich in Modulblättern befinden, ihre Anordnung ist belanglos.

Lösung der Aufgabe 3

Das Zeichnen des Drop-Down Listenfeldes im Summenblatt nach Schritt 1 ist rasch erledigt. Da wir auf dieses Objekt im Programmcode zugreifen wollen, geben wir ihm einen Namen. Wenn das Objekt markiert ist, kann in der Formelbearbeitungszeile, ganz links, der gewünschte Name eingegeben werden, z.B. `druckliste`. Im Code wird das Objekt dann mit `[druckliste]` angesprochen. (Für installierte Objekte gibt es auch interne Namen, die aber umständlich zu handhaben sind).

Mit der Methode `AddItem` wird ein Eintrag hinzugefügt. Er ist als Paramater in Form einer Textkette anzugeben.

In der Eigenschaft `value` des Listenfeldes ist die Nummer des Eintrages gespeichert, der zur Laufzeit mit der Maus gewählt wurde (0,1,2,..).

Die Methode `List(i)` schließlich gibt den Text des Eintrages mit der Nummer `i` zurück.

In einer For-Schleife können die Namen aller in der Anwendung vorhandenen Tabellenblätter in die Liste des Objektes `[druckliste]` aufgenommen werden.

```
' Drop-Down Listenobjekt [druckliste]
' in Tabelle ERGEBNIS mit Einträgen
' versorgen
  For i = 1 To Worksheets.Count
    Sheets("ERGEBNIS").[druckliste].AddItem Worksheets(i).Name
  Next i
```

Das Objekt aller Tabellenblätter, also die Modulblätter ausgenommen, wird mit der Methode `Worksheets` erzeugt. Die Eigenschaft `Count` gibt die Anzahl an. Die Methode `Worksheets(i)` gibt das Tabellenblatt mit der Nummer `i` zurück. In der Eigenschaft `Name` ist der Name gespeichert.

Damit die Listeneinträge entstehen, wenn die Anwendung geöffnet wird, müssen obige Anweisungen in der Prozedur `sub auto_open()` formuliert sein! Diese Prozedur kann in einem beliebigen Modulblatt stehen und kann nur ein mal vorkommen. (Modulblätter können mit dem Menüpunkt *Einfügen-Makro...-Visual Basic Modul* der Anwendung hinzugefügt werden.)

Als Ereignisprozedur für die Auswahl eines Eintrages in der Liste von `[druckliste]` schreiben wir in einem Modulblatt die Prozedur

```
Sub drucklisteauswahldrucken()
  Dim nr As Integer
  Dim obj As Object
  Dim tabname As String

  ' Zeiger zum Drop-Down Listenobjekt
  ' [druckliste] des
  ' Tabellenblattes ERGEBNIS in der
  ' Variablen obj speichern
   Set obj = Sheets("ERGEBNIS").[druckliste]
  ' Nummer des ausgewählten Eintrages
  nr = obj.Value
  ' Name des Eintrages feststellen
  tabname = obj.List(nr)
  tabdrucken (tabname)
End Sub
```

Nicht vergessen, diese Prozedur dem Drop-Down Listenfeld zuweisen!

Die Prozedur `tabdrucken` übernimmt ähnlich der Prozedur akttabdrukken das Ausdrucken des gewählten Tabellenblattes, nur wird der Name des Blattes als Parameter übergeben.

```
Sub tabdrucken(tname)
  Application.StatusBar = "Das  Tabellenblatt " & tname & "
wird gerade gedruckt."
  Sheets(tname).Select
  ActiveWindow.SelectedSheets.PrintOutCopies:=1
  Application.StatusBar = ""
End Sub
```

Lösung der Aufgabe 4

Die Bedienungselemente von Excel sind als Eigenschaften des Objekttes `Application` oder des Objektes `ActiveWindow` definiert. Z.B. deaktivieren die Anweisungen

```
Application.DisplayStatusBar = False
Application.DisplayFormulaBar = False
```

die Stauszeile und die Formelbearbeitungszeile.

Die Anzeige der Gitternetzlinien in einer Tabelle, der Scrollbalken, oder des Arbeitsblattregisters kann über Eigenschaften des Objektes `ActiveWindow` gesteuert werden. Die Anweisungen

```
Worksheets("Tabelle1").Select
ActiveWindow.Gridlines = False
ActiveWindow.DisplayHorizontalBar=False
ActiveWindow.DisplayWorkbookTabs=False
```

schalten die Gitternetzlinien, den horizontalen Scrollbalken und die Anzeige des Registers in ***Tabelle1*** ab.

Die Prozedur `symbol_aus()` deaktiviert alle aktiven Symbolleisten und speichert die Nummern der aktiv gewesenen Symbolleisten im Feld `toolsnr`. `symbol_aus` wird am besten in der Prozedur `auto_open()` aufgerufen. Damit beim Schließen der Anwendung der vorherige Zustand wieder hergestellt werden kann, muß die Prozedur `symbol_ein()` in der Prozedur `auto_close()` aufgerufen werden. `auto_close()` wird automatisch beim Schließen der Anwendung aufgerufen.

Die Prozeduren `koepfe_aus()`, `koepfe_ein()` und `status(wert)` stellen allgemein brauchbare Makros dar.

```
' Die Prozedur auto_close() wird beim
' Schließen der Anwendung automatisch ausgeführt
Sub auto_close()
  symbol_ein
End Sub

Sub koepfe_aus()
 Dim b As Integer
 For b = 1 To Worksheets.Count
  Worksheets(b).Select
  With ActiveWindow
   .DisplayGridlines = False
   ' Zeilenund Spaltenköpfe
   .DisplayHeadings = False
   .DisplayHorizontalScrollBar = False
   .DisplayVerticalScrollBar = False
    .DisplayWorkbookTabs = False
  End With
 Next b
End Sub

Sub symbol_aus()
  Dim i As Integer
  toolsakt = 0
  For i = 1 To Toolbars.Count
    If Toolbars(i).Visible = True Then
      Toolbars(i).Visible = False
      toolsakt = toolsakt + 1
      toolsnr(toolsakt) = i
    End If
  Next i
  'Statusleiste, Formelleiste
  status (False)
End Sub

Sub symbol_ein()
  Dim i As Integer
  For i = 1 To toolsakt
    Toolbars(toolsnr(i)).Visible=True
  Next i
  status (True)
End Sub

Sub koepfe_ein()
  Dim i As Integer
  For i = 1 To Worksheets.Count
    Worksheets(i).Select
    With ActiveWindow
      .DisplayGridlines = True
      .DisplayHeadings = True
      .DisplayHorizontalScrollBar=True
      .DisplayVerticalScrollBar = True
      .DisplayWorkbookTabs = True
    End With
  Next i
End Sub

Sub status(wert)
  With Application
    ' Formelbearbeitungszeile
    DisplayFormulaBar = wert
    ' Statuszeile
    DisplayStatusBar = wert
  End With
End Sub
```

Lösung der Aufgabe 5

Excel überläßt dem Programmierer nur eine geringe Anzahl von Ereignissen, auf die er mit einer selbst geschriebenen Prozedur, der zugehörigen Ereignisprozedur, reagieren kann.

Ereignisse werden als Eigenschaften von Objekten verwaltet. Die Namen der Eigenschaften sind mit `Onxyz` festgelegt und speichern den Namen der zugehörigen Ereignisprozedur als Textkette.

Für ein Tabellenblatt z.B. ist die Eigenschaft `OnEntry` definiert. Das Ereignis, das die in `OnEntry` gespeicherte Prozedur aufruft, ist die Veränderung des Inhaltes einer Zelle. Formatieren einer Zelle löst dieses Ereignis nicht aus, auch nicht von Programmanweisungen verursachte Inhaltsänderungen. Die Anweisung

```
Worksheets("KREIS").OnEntry="PruefeTab"
```

sieht vor, daß bei Zellinhaltsänderung im Tabellenblatt `KREIS` die Prozedur `PruefeTab()` aufgerufen wird. Leider können keine Übergabeparameter definiert werden. Paramaterübergaben, die z.B. steuern, welche Zellen nach welchen Gesichtspunkten zu testen sind, müssen über globale Variable geschehen. In unserer Anwendung wird für alle Tabellenblätter außer dem Summenblatt die Prozedur `PruefeTab()` als Ereignisprozedur für das Ereignis `OnEntry` festgelegt. Die einzelnen Anweisungen sind in der Prozedur `testi()` zusammengefaßt. Gleich darunter ist die Prozedur `PruefeTab()` geschrieben.

```
Sub testi()
Dim i As Integer

For i = 1 To Worksheets.Count
 If Worksheets(i).Name="ERGEBNIS" Then
  Worksheets(i).OnEntry = ""
  ' keine Datenprüfung !
 Else
  Worksheets(i).OnEntry = "PruefeTab"
 End If
Next i
End Sub

Sub pruefeTab()
  Dim bsp As Integer
  Dim esp As Integer
  Dim nr As Integer
  Dim z As Integer

  z = ActiveCell.Row
  ' Zeilennumer der aktiven Zelle
  If z <= 3 Or z >= 14 Then Exit Sub

  bsp = 2
  nr = tabnr
  'Nummer der aktiven Tabelle
  Select Case nr
    Case 2; 3; 7; 8; 9; 10: esp = 6
    Case 4; 5: esp = 5
    Case 6: esp = 8
  End Select

  With ActiveCell
    'Überprüfen der Werte in den
    'Spalten x,y,a,b und +-1
    If .Column >= bsp And .Column <= esp Then
      If IsNumeric(.Value) Then
      Else
        Beep
        MsgBox "Sie müssen eine Zahl eingeben !"
        If .Column = esp Then
```

```
          .Value=1
        Else
          .Value = ""
        End If
        Exit Sub
      End If
    End If
    'Überprüfen des Wertes in Spalte s bis (+-1)
    If .Column = esp Then
      If .Value ^ 2 <> 1 Then
        Beep
     MsgBox "Nur 1 oder -1 zulässig !"
        .Value = 1
      End If
    End If
  End With
  ' In Spalte "Nr." (1) die Nummer eintragen
  If IsEmpty(Cells(z; 1)) Then Cells(z; 1).Value = z-3
End Sub
```

Damit die Ereignisprozedur `PruefeTab()` beim Eintreten des Ereignisses tatsächlich aufgerufen wird, muß die Anweisung, in der die `OnEntry` Eigenschaft ihren Wert bekommt, in einer Prozedur stehen, die auch einmal aufgerufen wurde! Das entspricht dem Schritt 3, nämlich dem Zuweisen der Ereignisprozedur dem Steuerelement. Tatsächlich wird in diesem Fall intern die `OnAction`-Eigenschaft des Steuerelementes mit dem Prozedurnamen belegt! Aus diesem Grund wurde in unserer Anwendung die Prozedur `testi()` in die Prozedur `auto_open()` geschrieben.

Im Fehlerfall antwortet die Prüfroutine `PruefeTab()` mit einer Messagebox und einem entsprechenden Fehlertext (Figur 5).

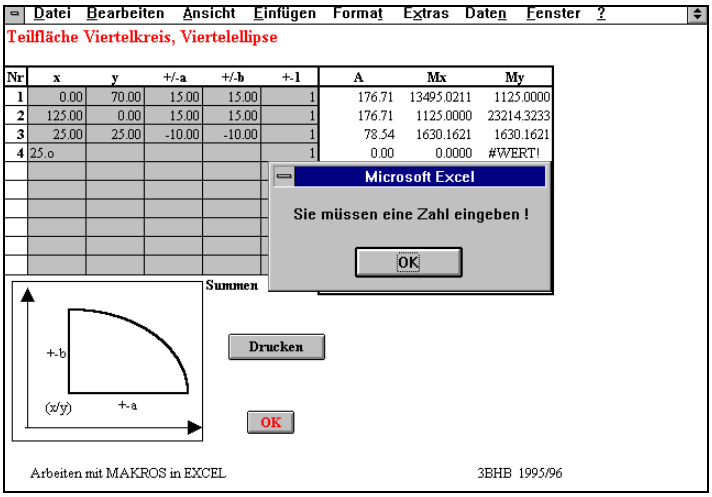

Abbildung 5

Weitere Aufgaben

Um Tabellen vor ungewollter Veränderung zu schützen, ist es sinnvoll, nur die vorgesehenen Datenbereiche für Eingaben zuzulassen und die anderen Bereiche zu schützen. Diese Aufgabe läßt sich (zeitraubend) über die Menüpunkte Format-Zellen...-Schutz und Extras-Dokument schützen... erreichen aber auch mit Excel-VBAEigenschaften und Methoden (`Protect, Unprotect, ..`).

Eine zusätzliche Schaltfläche im Summenblatt `ERGEBNIS` könnte veranlassen, daß nur jene Tabellenblätter gedruckt werden, in denen Daten eingegeben wurden.

Schließlich könnte das Beenden der Anwendung über eine Schaltfläche organisiert werden, so daß sichergestellt wird, daß die Anwendung mit den veränderten Blättern unter einem anderen Namen gespeichert wird, und die „leere“ Vorlage erhalten bleibt.

Schlußbemerkungen

Beschäftigt man sich erstmalig mit VBA für Excel, verläßt einen zunächst der Mut. Einmal ist VBA nicht gleich VB, was nicht besonders schwerwiegend ist, wenn man noch nicht mit VB programmiert hat. Es gibt in VBA zusätzliche Programmierelemente (die inzwischen in Visual Basic 4.0 aufgetaucht sind). Das Listentrennzeichen ist in VB das Komma (,), in VBA hängt es von Windows Systemeinstellungen ab (z.B. das Semikolon (;)).

Dazu kommt die unüberschaubare Menge an Excel VBA-Schlüsselwörtern (ca 2000), Objektnamen, Eigenschaftsnamen, Methodennamen etc. und die gewöhnungsbedürftige Objekthierarchie.

Nun zum Sprachproblem. Möchte man englische Schlüsselwörter verwenden, hat man seine liebe Not. Das installierte Hilfesystem ist meistens in Deutsch, im Handbuch findet man ebenfalls nur die eingedeutschten Schlüsselwörter. Die einzige Hilfe bietet die Datei `VBALISTE.XLS` im Verzeichnis `\EXCEL`, ein Wörterbuch, in dem alle VBA- und Excel VBA-Schlüsselwörter in Deutsch und Englisch gegenübergestellt sind.

Trotzdem ist Excel mit Visual Basic für Applikationen ein zukunftsweisendes Programmiersystem und könnte sogar reine Programmiersprachen verdrängen und ersetzen. □

AGTK 96043 - 19.02.1996 - Internet-Diskussion

****** Veraendert das INTERNET die Schule ? ********

Im Folgenden einige, zum Teil radikale Meinungen, zu denen in der Diskussion Stellung genommen werden soll. Zu bedenken ist, dass diese Thesen absichtlich so formuliert wurden, dass auch kontroversielle Stellungnahmen dazu provoziert werden.

1
Das INTERNET wird fuer Schulen so selbstverstaendlich werden wie heute Overheadprojektoren oder Computer.

2
Die Kommunikation im Internet wird dieselbe Bedeutung erlangen wie sie Konferenzen und Telephongespraeche heute schon haben.

3Informationen aller Art werden sich dynamisch im Netz verteilen und nicht mehr besorgt werden muessen.

4
Im Unterrichtsprozess wird Wissen nicht mehr vom Lehrer praesentiert sondern von allen Beteiligten eingebracht werden.

5
Die am Netz Beteiligten werden selbst das Informationsangebot steuern - die Gesellschaft bestimmt das Angebot.

6
Die Zugaenge zum Netz werden nicht zentral von Institutionen ermoeglicht, sondern werden individuell von Schulen gesucht werden. Ministerium und Schulbehoerde koennen den Weg ins Netz nicht mehr verordnen.

7
Die anfaengliche Spielphase wird nach einem Zurueckgehen der Nutzung einem gezielten Einsatz zu Unterrichtszwecken weichen.

8
Es stellt sich weniger die Frage "Wozu brauchen wir das?" sondern vielmehr die Aufgabe den Umgang mit der Informationsfuelle zu lernen.

9
Das INTERNET wird einer der wichtigsten Werbetraeger, auch fuer Schulen werden.

10
Es ist denkbar, dass das Internet ermoeglichen wird, die Zeit, die man in der Schule ist zu reduzieren.