16.6.3 Beispiele für den Einsatz einer TextBox – Validierung von Eingaben

Die Absicherung des Einsatzes valider Daten in einem Programm ist Teil des Konzeptes zur Vermeidung von Laufzeitfehlern. Im → Kapitel *16.6.2 Valide Daten* finden Sie Hinweise zum Thema und Ansätze, wie Sie Daten aus TextBoxen und ihren unterschiedlichen Spezialisierungen (*MaskBox*, *ValueBox*, *HistoryBox, InputBox* und *ButtonBox) v*alidieren können. Für die Eingabe von komplexen Zahlen, Matrizen, Polynomen oder Vektoren dagegen finden Sie Beispiele in den → Kapiteln 29.1 Komplexe Zahlen, 29.3.3 Matrix, 29.3.4 Polynom und 29.3.5 Vector.

Für die Prüfung, ob sich ein Eingabe-String aus einer TextBox in ein valides Datum konvertieren lässt – wenn die Spezialisierungen nicht verwendet werden können – gibt es kein allgemein gültiges Verfahren. Die folgenden Beispiele sollen Ihnen aber zeigen, wie Sie mit unterschiedlichen Strategien einem Programm valide Daten aus einem Eingabe-String zur Verarbeitung bereit stellen.

16.6.3.1 Beispiel 1

In einer TextBox werden eine Anzahl A erfasst und in einer weiteren TextBox ein Messwert-Mittelwert (ohne Einheit) als reelle Zahl R. Aus den beiden Eingaben wird im Programm ein Quotient mit vier Dezimalstellen berechnet QM = (A²-Pi) / R und in einer dritten TextBox ausgegeben (read only). Bei jeder Änderung der Eingaben wird der Inhalt der Ausgabe-TextBox gelöscht.

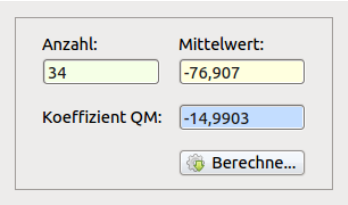

Abbildung 16.6.3.1.1: Berechnung QM

Bevor Sie zügig loslegen, sollten Sie folgende Hinweise in aller Ruhe durchdenken, wobei die Reihenfolge keiner Rangfolge entspricht:

- Die Eingaben in den TextBoxen sind vom Datentyp String. Für die Berechnung des Quotienten benötigen Sie aber Zahlen.
- Bei der Eingabe in die beiden TextBoxen sollten Sie das Eingabe-Alphabet aufgaben-adäquat einschränken.
- Während für die Eingabe der Anzahl als Zeichen nur einzelne Ziffern zulässig sind, kann das Eingabe-Alphabet für die reelle Zahl so notiert werden: {+-,0123456789}.
- Die Festlegung des 2. Alphabets zum Beispiel schützt in keiner Weise vor völlig sinnlosen Eingaben wie '+0,874-,9', was Sie zur Validierung der Daten zwingt. Sie müssen prüfen, ob sich der Eingabe-String aus der TextBox für den Messwert-Mittelwert R *sicher* in eine reelle Zahl konvertieren lässt.
- Die Konvertierung des in unserem Sprachraum gewohnten Kommas in einen Punkt müssen Sie programm-intern realisieren.
- Die Anzahlen – gespeichert in einer geeignet genannten Variable – müssen Sie in den Datentyp Integer konvertieren, wobei hier festgelegt wurde, die Anzahl 0 nicht als valides Datum zuzulassen, was zu prüfen ist. Auf das Eingabe-Alphabet {0123456789} hat das jedoch *keine* Auswirkungen, denn sonst wäre zum Beispiel die Anzahl 10 unmöglich.
- Der Messwert R ist in einer Variablen vom Datentyp Float gut aufgehoben. Die Konvertierung in diesen Typ muss dem Rechnung tragen.
- Gambas stellt Ihnen mit *CFloat(Value As Variant)* eine geeignete Funktion für die Konvertierung zur Verfügung.
- Um bei der Berechnung des Quotienten keinen Fehler 'Division durch Null' zu provozieren, ist die Eingabe einer Null auszufiltern und nach einem Hinweis an den Benutzer eine Korrektur für den fehlerhaften Nenner anzubieten.
- Bevor Sie die Berechnung anschieben, sollten Sie prüfen, ob die Eingabe-TextBoxen nicht leer sind. Für diesen Fall ist der Programmablauf zu unterbrechen, der Benutzer in geeigneter Weise zu informieren und Korrekturmöglichkeiten zu offerieren.

Der Quelltext wird auszugsweise vorgestellt:

```
[1]  ' Die Eigenschaft txbResult.ReadOnly wird auf TRUE gesetzt
[2]  Public Sub btnCalculate_Click()
[3]    Dim sInputN, sInputR, sPatternN, sPatternR As String
[4]    Dim iCount As Integer
[5]    Dim fValue, fCoefficient As Float
[6]
[7]    If Not txbInteger.Text Then ' Prüfung: TextBox leer?
[8]       Message.Error("Eingabefehler 1!") ' Kommentar in einer MessageBox
[9]       txbInteger.SetFocus ' Fokus auf die TextBox txbInteger setzen
[10]      Return ' Prozedur verlassen
[11]   Endif
[12]   If Not txbReal.Text Then ' Prüfung: TextBox leer?
[13]      Message.Error("Eingabefehler 2!")
[14]      txbInteger.SetFocus
[15]      Return
[16]   Endif
[17]
[18]  '------------------------------------------------------------------------
[19]
[20]   sInputN = Trim(txbInteger.Text) ' Leerzeichen am Anfang und am Ende entfernen
[21]   Try iCount = CInteger(sInputN) ' Versuch: Konvertierung String → Ganze Zahl
[22]   If Error Then ' Fehlerbehandlung
[23]      Message("Fehler 1")
[24]      txbInteger.SetFocus
[25]      Return
[26]   Endif
[27]
[28] ' Anzahl Null herausfiltern
[29]   If iCount = 0 Then
[30]      Message.Info("Die Anzahl muss größer als Null sein!")
[31]      txbReal.SetFocus
[32]      Return
[33]   Endif
[34]
[35] '------------------------------------------------------------------------
[36]
[37]   sInputR = Trim(txbReal.Text) ' Leerzeichen am Anfang und am Ende entfernen
[38] ' Dezimalseparator der aktuellen Locale durch den Punkt ersetzen
[39]   sInputR = Replace$(sInputR, Left$(Format$(0, ".0")), ".")
[40]
[41]   Try fValue = CFloat(sInputR) ' Versuch: Konvertierung String → Reelle Zahl
[42]   If Error Then
[43]      Message("Fehler 2")
[44]      txbReal.SetFocus
[45]      Return
[46]   Endif
[47]
[48]   If fValue = 0 Then ' Null herausfiltern
[49]      Message.Info("Die (reelle) Zahl 0 ist nicht zulässig!")
[50]      txbReal.SetFocus
[51]      Return
[52]   Endif
[53]
[54] '------------------------------------------------------------------------
[55]
[56]   fCoefficient = Round((iCount * iCount - Pi()) / fValue, -4) ' Berechnung Quotient
[57]   txbResult.Text = Str(fCoefficient) ' Anzeige Quotient (Koeffizient QM)
[58]
[59] End ' btnCalculate_Click()
[60]
[61] '------------------------------------------------------------------------
[62]
[63] Public Sub txbInteger_KeyPress() ' Umsetzung Eingabe-Alphabet
[64]   CheckInput("0123456789")
[65] End ' txbDigits_KeyPress()
[66]
[67] Public Sub txbReal_KeyPress() ' Umsetzung Eingabe-Alphabet
[68]   CheckInput("+-,0123456789")
[69] End ' txbDigits_KeyPress()
[70]
[71] Public Sub CheckInput(sAllowed As String) ' Idee Charles Guerin
[72]   Select Case Key.Code
[73]     Case Key.Left, Key.Right, Key.BackSpace, Key.Delete, Key.End, Key.Home, Key.Enter, Key.Return
[74]       Return
[75]     Default
[76]       If Key.Text And If InStr(sAllowed, Key.Text) Then
[77]          Return
[78]       Endif
[79]   End Select
[80]   Stop Event
[81] End ' CheckInput(sAllowed As String)
```

```
[82]
[83] ' Public Sub txbInteger_Change()
[84] '    txbResult.Clear
[85] ' End
[86] ' Public Sub txbReal_Change()
[87] '   txbResult.Clear
[88] ' End
[89]
[90] Public Sub TBIR_Change()
[91]   txbResult.Clear
[92] End ' TBIR_Change()
```

Kommentar:

- In den Zeilen 83 bis 88 wird die Forderung realisiert, dass bei jeder Änderung der Text-Eigenschaft in den beiden TextBoxen *txbInteger* und *txbReal* der Text in der Ausgabe-TextBox gelöscht wird. Für zwei TextBoxen ist der Aufwand noch vertretbar – aber bei 5 oder mehr? Dann lohnt es sich, bei allen beteiligten TextBoxen die *virtuelle Eigenschaft Group* in der IDE jeweils auf den gleichen Wert – hier TBIR – zu setzen und für alle einen gemeinsamen Event-Handler zu definieren. Die Implementation für die beiden TextBoxen *txbInteger* und *txbReal* sehen Sie in den Zeilen 83 bis 85.
- Die Hinweistexte und Fehlermeldungen können Sie noch spezifisch formulieren.
- Eine Vereinfachung ergäbe sich in den Zeilen 20 bis 33 durch den Einsatz eines regulären Ausdrucks, da durch die Musterprüfung Null als Muster nicht akzeptiert wird und nur korrekte Anzahlen eingegeben werden können:

```
[1]    sInputN = Trim(txbInteger.Text)
[2]    sPatternN = "^[1-9][0-9]*$" ' Alternative mit Null: "^[0]{1}$|^[1-9][0-9]*$"
[3]    If sInputN Not Match sPatternN Then
[4]       Message.Error("Eingabefehler 3!")
[5]       txbInteger.SetFocus
[6]       Return
[7]    Endif
[8]    iCount = CInteger(sInputN)
```

16.6.3.2 Beispiel 2

Damit nur bestimmte Zeichen – hier sind es die Ziffern 0 bis 9 – in einer TextBox akzeptiert werden, wird in der Dokumentation der folgende Beispiel-Quelltext offeriert:

```
' My text box only accepts digits

Public Sub MyTextBox_KeyPress()

  If Instr("0123456789", Key.Text) = 0 Then
     Stop event
  Endif

End
```

Hinweise:

- In der Gambas-Dokumentation wird beschrieben, dass man mit dem *Stop Event* dem Interpreter mitteilt, dass das aktuelle Event 'gecancelt' werden soll.
- Manche Ereignisse unterstützen die Aktion, abgebrochen zu werden. Das KeyPress-Ereignis gehört dazu. Wenn es abgebrochen wird, dann wird der Tastendruck nicht weiter verarbeitet – die Eingabe wird verworfen. Dabei ist es egal, wo *Stop Event* innerhalb des Event-Handlers steht. Es ist aber zwingend erforderlich, um das Ereignis abzubrechen.

Funktioniert – doch leider können Sie die Ziffernfolge in keiner Weise korrigieren, was als Mangel angesehen werden kann. Deshalb wird im folgenden Abschnitt eine verbesserte Lösung vorgestellt, bei der Sie den Cursor in der TextBox bewegen und mit der BackSpace-Taste einzelne Ziffern löschen können:

```
Public Sub txbDigits_KeyPress()
' Nur die Ziffern 0..9 als zulässige Ziffern zulassen
  If (Key.Text Not Like "[0-9]") And (Key.Code <> Key.BackSpace) And (Key.Code <> Key.Left) \\
     And (Key.Code <> Key.Right) Then
     Stop Event
  Endif ' Key.Text Not Like "[0-9]" ?
End ' txbDigits_KeyPress()
```

Einen Nachteil werden Sie sicher noch entdecken – das Eingabe-Alphabet ist nicht variabel, wenn Sie beispielsweise an den Fall denken, dass alle Ziffern 0-9 zulässig sind sowie *zusätzlich* die beiden Vorzeichen (+-) und die Zeichen Punkt, Komma und Sternchen wie auch der Buchstabe x. Die folgende Lösung behebt den Nachteil sicher:

```
Public Sub CheckInput(sAllowed As String) ' Idee Charles Guerin
  Select Case Key.Code
    Case Key.Left, Key.Right, Key.BackSpace, Key.Delete, Key.End, Key.Home, Key.Enter, Key.Return
      Return
    Default
      If Key.Text And If InStr(sAllowed, Key.Text) Then
         Return
      Endif
  End Select
  Stop Event

End ' CheckInput(sAllowed As String)

Public Sub txbDigits_KeyPress()
  CheckInput("+-,.*0123456789x")
End ' txbDigits_KeyPress()
```

Die Prüfung, ob ein eingegebenes Zeichen zum Eingabe-Alphabet gehört oder nicht, arbeitet fehlerfrei. Denken Sie aber daran, den Eingabe-String anschließend noch einer Validitätsprüfung zu unterziehen, wenn zum Beispiel der Eingabe-String als Term in einer linearen Gleichung aufgefasst werden soll. Der String *-5,88*x+7,1+*23* enthält nur Zeichen aus dem Eingabe-Alphabet und ist trotzdem für die Verarbeitung nicht geeignet, da er einen Syntax-Fehler im Sinne der Term-Vorgabe enthält.

16.6.3.3 Beispiel 3

Dieses Beispiel setzt nicht auf Prüfung sondern auf Prävention, weil dem Benutzer ein Hinweis-Text in der TextBox angezeigt wird, um Fehleingaben zu minimieren. Genutzt wird eine spezielle von *Tobias Boege* entwickelte TextBox – die Explain-TextBox. Die Explain-TextBox ist eine TextBox, die es erlaubt, einen erklärenden Vorgabe-Text (Explanation) in der Schriftfarbe grau anzuzeigen, wenn die TextBox *leer* ist und nicht den Fokus hat. Damit könnten Sie u.U. auf erklärende Beschriftungen vor den TextBoxen verzichten:

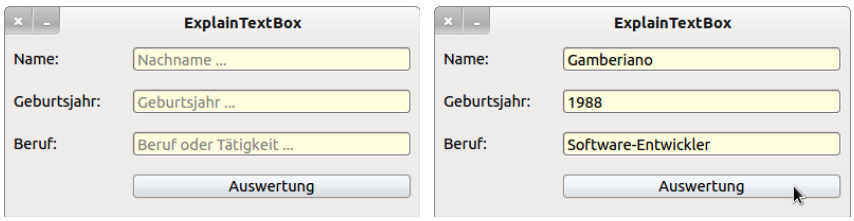

Abbildung 16.6.3.3.1: Explain-TextBox

Die Klasse *ExplainTextbox* wird Ihnen in einem Projekt im *Download-Bereich* zur Verfügung gestellt.

16.6.3.4 Beispiel 4

Es soll geprüft werden, ob es sich bei der Eingabe einer Jahreszahl im Intervall von 1600 bis 2100 in eine TextBox um ein Schaltjahr handelte oder handeln wird.

Ein astronomisches Jahr ist die Dauer der Revolution der Erde um den Zentralkörper Sonne – unseren Stern. In unseren Zeiteinheiten sind das sekundengenau 365 Tage, 5 Stunden, 48 Minuten und 47 Sekunden. Ein normales Kalender-Jahr – aber was ist schon normal – hat eine Länge von 365 Tagen. Nach 4 Jahren werden die 5 Stunden, 48 Minuten und 47 Sekunden, was ja in etwa einem Viertel-Tag entspricht, zu einem Tag, dem 29. Februar, zusammengefasst und damit hat ein Schaltjahr 366 Tage. Ein Schaltjahr gibt es alle 4 Jahre, jedoch alle 100 Jahre nicht, dann aber alle 400 Jahre doch, um die Abweichungen der eingefügten Viertel-Tage zu 6 Stunden auszugleichen. Ich bin begeistert!

Als vorbeugende Maßnahme wird die Anzahl der Zeichen auf maximal 4 Zeichen begrenzt:

```
[1]  Public Sub Form_Open()
[2]    txbLeapYear.MaxLength = 4
[3]    ...
[4]  End ' Form_Open()
```

Die Funktion IsLeapYear(..) setzt die o.a. Regel für Schaltjahre um und gibt den Funktionswert *True* zurück, wenn das als Argument eingesetzte Jahr ein Schaltjahr war oder ist:

```
[1]  Public Function IsLeapYear(iYear As Integer) As Boolean
[2]
[3]    If (iYear Mod 4) = 0 And (iYear Mod 100) <> 0 Or ((iYear Mod 400) = 0) Then
[4]       Return True
[5]    Else
[6]       Return False
[7]    Endif
[8]
[9]  End ' Function IsLeapYear(..)
```

Im folgenden Quelltext-Abschnitt – dem ein Kommentar folgt – sehen Sie diverse (Vor-)Prüfungen:

```
[1]  Public Sub txbLeapYear_Activate()
[2]    Dim iCount As Integer
[3]    Dim sInput As String
[4]
[5]    sInput = Trim(txbLeapYear.Text) ' Entfernung von Leerzeichen am Anfang und am Ende des Strings
[6]
[7]    For iCount = 1 To Len(sInput)
[8]        If Mid(sInput, iCount, 1) Not Like "[1234567890]" Then ' Zeichen-Prüfung auch mit [0-9]
[9]           Message.Error("Eingabefehler - Zeichen!")
[10]          Return
[11]       Endif
[12]    Next
[13]
[14]   If Len(sInput) <> 4 Then
[15]      Message.Error("Jahreszahl nicht vierstellig!") ' optional
[16]      Return
[17]   Endif
[18]
[19]   If CInt(sInput) < 1600 Or CInt(sInput) > 2100 Then ' Bereichsprüfung [1600..2100]
[20]      Message.Error("Eingabefehler - Bereich!")
[21]      Return
[22]   Endif
[23]
[24]   If IsLeapYear(CInt(sInput)) Then ' Prüfung Schaltjahr und Kommentare
[25]      If CInt(sInput) < Year(Now) Then
[26]         Message.Info("Das Jahr " & sInput & " war ein Schaltjahr.")
[27]      Else
[28]         Message.Info("Das Jahr " & sInput & " wird ein Schaltjahr sein.")
[29]      Endif
[30]   Else
[31]      If CInt(sInput) < Year(Now) Then
[32]         Message.Info("Das Jahr " & sInput & " war kein Schaltjahr.")
[33]      Else
[34]         Message.Info("Das Jahr " & sInput & " wird kein Schaltjahr sein.")
[35]      Endif
[36]   Endif
[37]
[38] End ' txbLeapYear_Activate()
```

Kommentar:

- In der Zeile 5 wird der Variablen *sInput* die Zeichenkette zugewiesen, die in der TextBox steht, wenn die Texteingabe mit der Return-Taste abgeschlossen wurde.
- In den Zeilen 7 bis 12 erfolgt in einem Zeichen-Scanner die Prüfung, ob alle Zeichen im Eingabe-String Ziffern (0..9) sind, nachdem Leerzeichen entfernt worden sind.
- Wurden nicht genau 4 Ziffern eingegeben – Zeilen 14 bis 17 – so erfolgt eine entsprechende Mitteilung und Sie können Korrekturen vornehmen (Option).
- Eine Bereichsprüfung auf den vorgegebenen Zeitraum wird in den Zeilen 19 bis 22 vorgenommen.
- Da das Argument der Funktion *IsLeapYear(iYear AS Integer)* (Zeile 24) vom Daten-Typ eine ganze Zahl sein muss, wird der Eingabe-String mit CInt(..) in eine ganze Zahl konvertiert, was nach den erfolgreichen Vorprüfungen auch problemlos möglich ist.

- Je nach Funktionswert werden detaillierte Mitteilungen zum Ausgang der Prüfung angezeigt.

Die hier vorgestellte Alternative nutzt einen *regulären Ausdruck*, mit dem sämtliche Prüfungen *in einem Schritt* erfolgen:

```
Public Sub txbLeapYear_Activate()
  Dim sSubject, sPattern As String

  sSubject = Trim(txbLeapYear.Text)
  sPattern = "^([1][6789][0-9]{2})|[2][0]([0-9]{2})|2100$" ' Prüf-Muster als regulärer Ausdruck

  If sSubject Not Match sPattern Then
     Message.Error("Eingabefehler!")
     Return
  Endif

  Message.Info(Subst$("Das Jahr &1 ist &2ein Schaltjahr!",sSubject,IIf(IsLeapYear(CInt(sSubject)),"","k")))

End ' txbLeapYear_Activate()
```

16.6.3.5 Beispiel 5

Mit dem folgenden Quelltext können Sie die beiden TextBoxen *txbA* und *txbB* bei der Eingabe komplett synchronisieren:

```
Public Sub txbA_Change()
  txbB.Text = txbA.Text
End ' txbA_Change()
```

16.6.3.6 Beispiel 6

In diesem Activate-Ereignis ersetzen Sie ein Komma – so wie Sie es zum Beispiel bei der Eingabe von Dezimalzahlen gewohnt sind – intern durch einen Punkt, so dass nach einer Validitätsprüfung des Eingabe-Strings dem Programm eine Dezimalzahl zur Verarbeitung übergeben werden kann. Probieren Sie es aus:

```
Public Sub txbInput_Activate()
  Dim sInput As String = txbInput.Text

' Dezimalseparator der aktuellen Locale durch den Punkt ersetzen
  Print sInput
    sInput = Replace$(sInput, Left$(Format$(0, ".0")), ".")
  Print sInput

End ' txbInput_Activate()
```

16.6.3.7 Beispiel 7

Auf einem Formular existieren n TextBoxen, bei denen von k ausgewählten TextBoxen (k ≤ n) der *Inhalt* mit einem Aufruf gelöscht werden soll. Die Lösungsidee besteht darin, für die TextBoxen, deren Inhalt zu löschen ist, ein Alleinstellungsmerkmal zu definieren und dafür bietet sich zum Beispiel die *Tag-Eigenschaft* an.

```
Public Sub Form_Open()
  FMain.Center

  txbBox1.Tag = "L"
  txbBox2.Tag = "L"
  txbBox4.Tag = "L"
  txbBox6.Tag = "L"
  ...
End ' Form_Open()

Private Sub TBMultiClear()
  Dim C As Control
  Dim tBox As TextBox
  For Each C In Me.Controls
     If C Is TextBox Then
        If C.Tag = "L" Then
         ' Print C.Name ' Kontrolle
           tBox = C
           tBox.Clear
        Endif
     Endif
  Next
End ' TBMultiClear()

Public Sub btnTBClear_Click()
  TBMultiClear()
End ' btnTBClear_Click()
```

16.6.3.8 Beispiel 8

Folgende Aufgabe ist zu bearbeiten: Bei der Auslösung des Activate-Events einer Textbox A (Druck auf die Return-Taste) soll eine bestimmte Aktion ausgelöst werden und danach der Cursor in die TextBox B gesetzt werden, die den Fokus erhält. Von Hand geht das mit der Tabulatortaste, wenn die TextBox B in der Hierarchie *unmittelbar* der TextBox A folgt. Die Hierarchie von Komponenten legen Sie in der IDE im Hierarchie-Fenster fest. Der folgende Quelltext realisiert das oben genannte Verhalten *ebenso*:

```
Public Sub txb_A_KeyPress()
  If Key.Code = Key.Enter Or Key.Code = Key.Return Then
   ' AKTION ...
     txb_B.SetFocus()
   ' Desktop.SendKeys("\t") ' Alternative, wenn die Komponente Desktop verwendet wird
  Endif
End ' txb_A_KeyPress()
```

Die Verwendung der Alternative hat den Vorteil, dass der Text in der TextBox B markiert wird, wie das auch bei der Verwendung der Tabulatortaste erfolgt.