

Eingabe numerischer Werte in Pascal-Anwendungen

Ein Vorschlag

In vielen Programmen ist die Eingabe von Zahlenwerten durch die User notwendig. Die sollte möglichst einfach und nutzungsfreundlich gestaltet werden. Die Eingabe von falschen Zeichen (Buchstaben statt Zahlen) ist nach Möglichkeit zu verhindern. Besonders leidige Erfahrungen haben viele Softwareentwickelnde mit – na klar – Punkt und Komma erlitten.

Sobald die Software z. B. das Kommagebiet (räumliches Gebiet mit Nutzung des Komma als Dezimaltrennzeichen) verlässt und in einem Punktgebiet eingesetzt wird, kann das die Nutzung der Software unmöglich machen.

Auch die Weitergabe von Quelltexten kann zu solchen Problemen führen.

Ganz schlimm ist es, mit einem "deutschen" Rechner in China Software zu entwickeln und die dann auf chinesische Rechner zu bringen. Deshalb soll hier ein Vorschlag gezeigt werden, wie diese Probleme bei der Softwareentwicklung mit Free-Pascal und Lazarus vermieden werden können.

Das ursprüngliche Pascal kennt als Dezimaltrennzeichen ausschließlich den Punkt. Daher wird hier der Punkt als Standard gewählt.

Die Komponenten TEdit (Tab Standard) und TLabelEdit (Tab Additional) sind allgemeine Komponenten zur Eingabe von beliebigen Zeichen – Entwickelnde müssen selbst dafür sorgen, dass die Eingaben nicht zum Programmabsturz führen.

Nutzbar sind weitere Standardkomponenten, welche in der Komponentenpalette zu finden sind:

- **TMaskEdit:** Die möglichen Eingaben können durch Masken gesteuert werden. Komplizierte bis unverständliche Komponente (Zu finden im Tab Additional).
- **TSpinEdit:** Eingabezeile für Integer mit up / down - Pfeilen am rechten Rand, welche ein Hoch- oder Runterzählen mit der Maus ermöglichen. (Zu finden im Tab Misc).
- **TFloatSpinEdit:** Wie TSpinEdit aber für Floats. (Zu finden im Tab Misc).
- **TEditButton:** Eine TEdit mit integriertem Speed-Button. Zu finden im Tab Misc).
- **TSpinEditEx:** Wie die TSpinEdit, die Pfeile können ausgeblendet werden. Zu finden im Tab LazControls).
- **TFloatSpinEditEx:** Wie die TFloatSpinEdit, die Pfeile können ausgeblendet werden. Zu finden im Tab LazControls).

Ganz gut ist TSpinEdit: Es wird tatsächlich nur die Eingabe von Integer zugelassen, außer dem Minus sind nur Zahlen zugelassen. Schade ist, dass das Minus auch zwischen zwei Zahlen gesetzt werden kann, die Rückgabe ist dann 0.

TFloatSpinEdit ist auf halben Weg eingeknickt: Ob Komma oder Punkt eingetippt werden, es wird ein Punkt angezeigt und auch als Zahl nachher verarbeitet. Das ist gut – jedoch ist die Eingabe mehrerer Punkte möglich, was dann wieder zu einer ungültigen Eingabe führt.

Als Buchstabe ist nur ein 'e' für Zehnerpotenzen möglich, das ist gut.

Minus- und Pluszeichen sind wieder mehrfach möglich, das ist nicht so gut und ergibt dann wieder die Null als Resultat.

Irgendwo ist das alles nicht konsequent, daher hier eine Unit mit den Prozeduren und Funktionen

- function Str2Float(aStr: string): extended;
- function Str2Int(aStr: string): int64;
- procedure MyIntInput(Sender: TObject; var Key: char);
- procedure MyFloatInput(Sender: TObject; var Key: char);
- function Int2Str(aInt: int64; nPlaces: integer): string;
- function Float2Str(aFloat: extended; nPlaces, nDecimals: integer): string;

Notwendig sind die Funktionen Str2Int und Str2Float, weil StrToInt und StrToFloat je nach Gebietsschema Punkte oder Kommata erwarten. Außerdem sollt die Eingabe langer Zahlen durch Formatierungsmöglichkeiten erleichtert werden. So können Leerzeichen oder Unterstriche zur formatierten Eingabe verwendet werden.

Float2Str und Int2Str wurden erstellt, um die Zahlen mit einer definierten Anzahl von Stellen ausgeben zu können. Dadurch ist auch in Textdateien eine tabellarisch wirkenden Ausgabe möglich, wenn diese mit einem entsprechendem Zeichensatz erfolgt.

In der Beispielanwendung werden für die Eingabe TEdit und TLabelEdit verwendet und die Eingaben nach dem Klick auf "ok" in einem TMemo ausgegeben, Bild 1.

Input im TEdit-Input: -50.0112300000
Input im TLabelEdit-Input: 123.4000000000
Integer input im TEdit-Input: 42
Integer input im TEdit-Input: 123456
Integer input im TEdit-Input: 112
Ausgabe der Integer mit StrToInt und IntToStr:
Integer input im TEdit-Input: 42
Integer input im TEdit-Input: 123456
Integer input im TEdit-Input: 112
Input im FloatSpinEditEx: 42.42
Input im SpinEditEx: 42

Bild 1: Beispielanwendung mit unterschiedlichen Eingaben und Ausgaben

Die Eingabe wird über das Ereignis "OnKeyPress" zu "FloatInput" bzw. "IntInput" umgeleitet und dann überprüft und gegebenenfalls geändert:

- Für Integer sind nur Zahlen, das Minuszeichen und ein Punkt sowie Leerzeichen oder Unterstriche zum Formatieren möglich.
 - Ein Komma wird in einen Punkt geändert.
 - Die Eingabe von "123_456_789" oder "123 456 789" wird im Str2Int zu Integerzahl "123456789" geändert.
 - Die Eingabe eines zweiten Punktes oder eines zweiten Minuszeichens werden unterbunden.
- Für Floats sind Zahlen, Minuszeichen, das 'e' und Leerzeichen oder Unterstrich zum Formatieren möglich.
 - Ein Komma wird in einen Punkt geändert.

- Die Eingabe eines zweiten 'e' wird unterbunden.
- Ist kein 'e' im Eingabefeld, wird die Eingabe eines zweiten Minuszeichens unterbunden.
- Ist ein 'e' im Eingabefeld vorhanden, dann
 - sind zwei Minuszeichen möglich: Eins vor der Zahl, eins hinter dem 'e';
 - wird Eingabe eines dritten Minuszeichens unterbunden.
- "-123 456 789 e -2" oder "123_456_789_e_-2" ergeben "-1234567.89".

Im Anhang sind die Flowcharts zur procedure MyFloatInput“ und der darin enthaltenen procedure HandleTEdit zu finden. Die procedure ThangleLabeledEdit ist genauso aufgebaut. Für die procedure MyIntInput ist wegen ihrer Einfachheit kein Flowchart erforderlich.

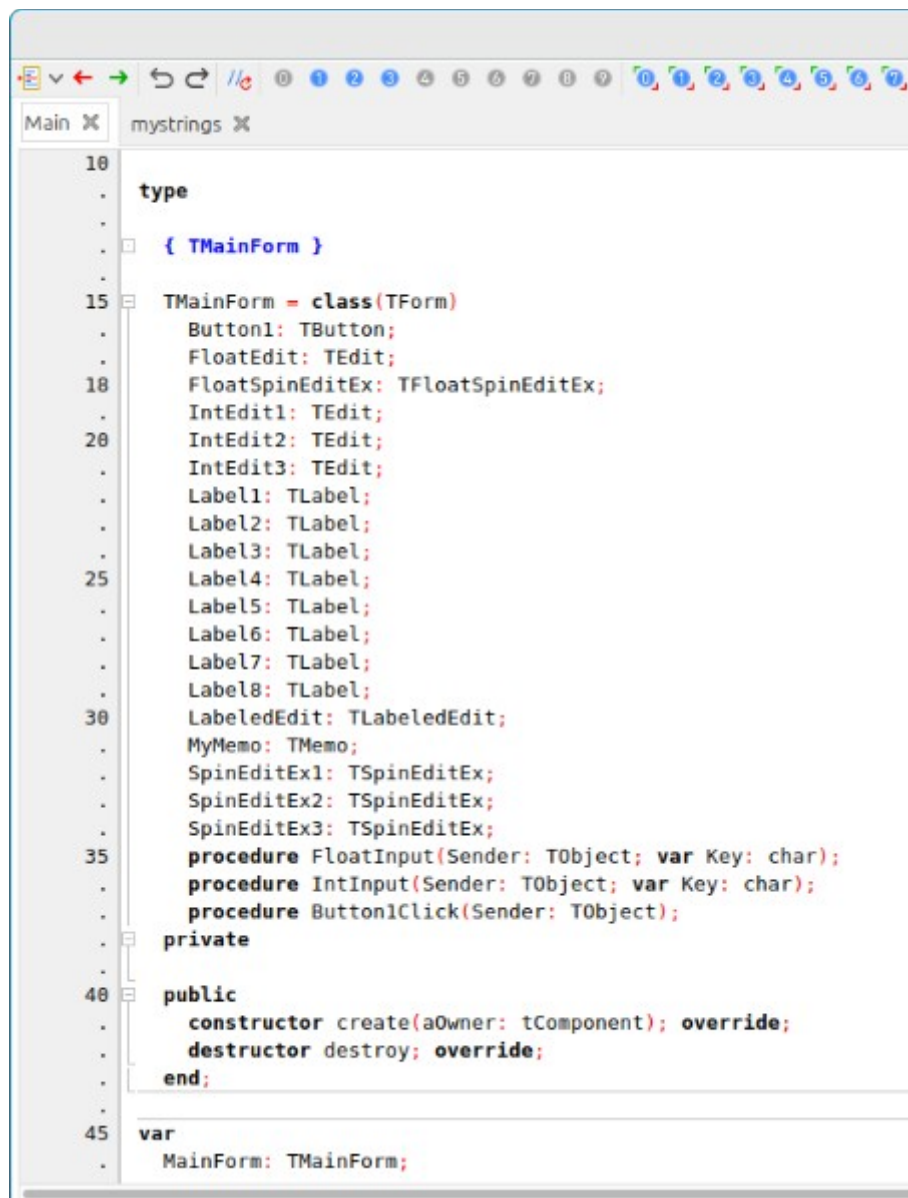
Der Quelltext ist sicher ausreichend, um das Vorgehen zu verstehen. Das Verbinden vom Ereigniss des Tastendrucks und die Definition der Prozeduren sind im Anhang aber noch zusätzlich bebildert aufgeführt.

Fazit

Ganz klar: Die User können durch die Eingabe unsinniger Zahlen viele Programme in "Schwierigkeiten" bringen. Es ist Aufgabe der Entwicklung, dies nach Möglichkeit zu verhindern. Die Eingabe von Zahlen auf die möglichen Zeichen sowie deren Position auf die sinnvollen Positionen zu begrenzen, stellt keine Bevormundung dar, sondern eine Erleichterung. Sowohl für die User als auch für die Entwicklung.

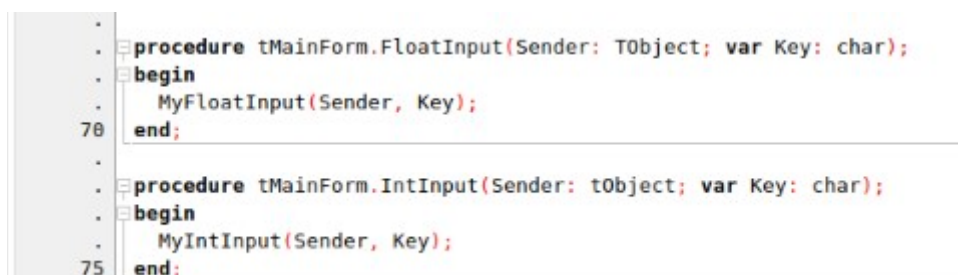
Anhang

Prozeduren für die Input-Ereignisse erstellen



```
10 . type
.
. { TMainForm }
.
15 TMainForm = class(TForm)
.   Button1: TButton;
.   FloatEdit: TEdit;
18   FloatSpinEditEx: TFloatSpinEditEx;
.   IntEdit1: TEdit;
20   IntEdit2: TEdit;
.   IntEdit3: TEdit;
.   Label1: TLabel;
.   Label2: TLabel;
.   Label3: TLabel;
25   Label4: TLabel;
.   Label5: TLabel;
.   Label6: TLabel;
.   Label7: TLabel;
.   Label8: TLabel;
30   LabelledEdit: TLabelledEdit;
.   MyMemo: TMemo;
.   SpinEditEx1: TSpinEditEx;
.   SpinEditEx2: TSpinEditEx;
.   SpinEditEx3: TSpinEditEx;
35   procedure FloatInput(Sender: TObject; var Key: char);
.   procedure IntInput(Sender: TObject; var Key: char);
.   procedure Button1Click(Sender: TObject);
. private
.
.
40 public
.   constructor create(aOwner: TComponent); override;
.   destructor destroy; override;
. end;
.
45 var
.   MainForm: TMainForm;
```

Bild 2: Definition von TMainForm



```
.
. procedure tMainForm.FloatInput(Sender: TObject; var Key: char);
. begin
.   MyFloatInput(Sender, Key);
70 end;
.
. procedure tMainForm.IntInput(Sender: TObject; var Key: char);
. begin
.   MyIntInput(Sender, Key);
75 end;
```

Bild 3: Aufruf der Prozeduren aus der Unit MyStrings.pas

Ereignisse mit Prozeduren verbinden

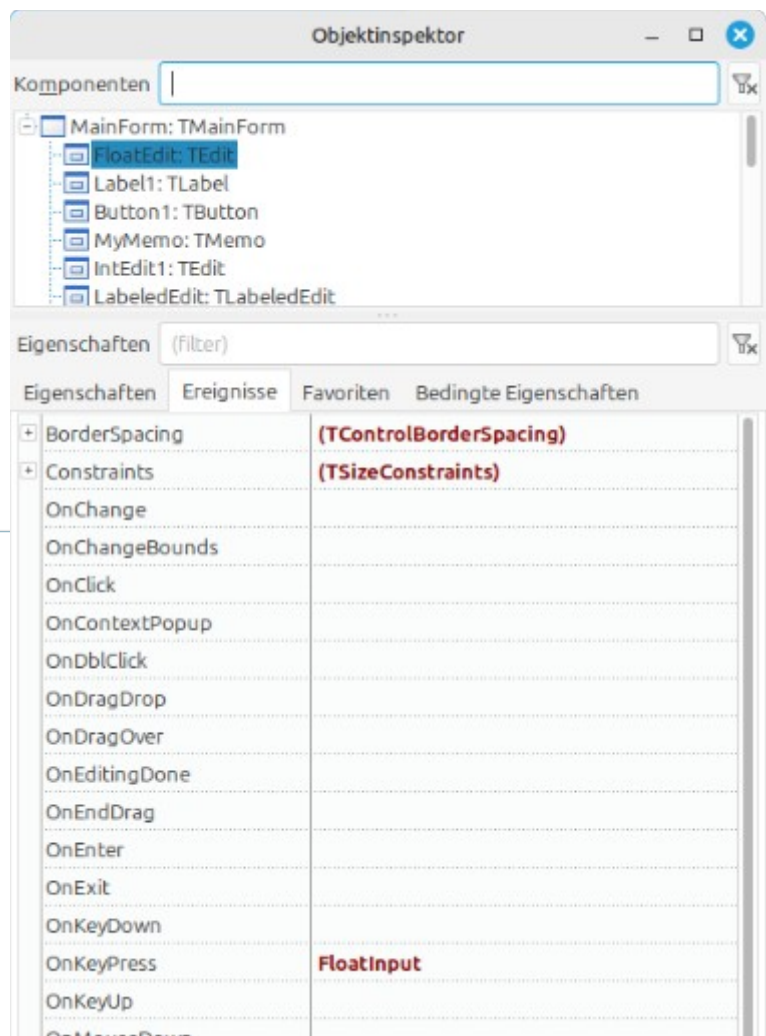
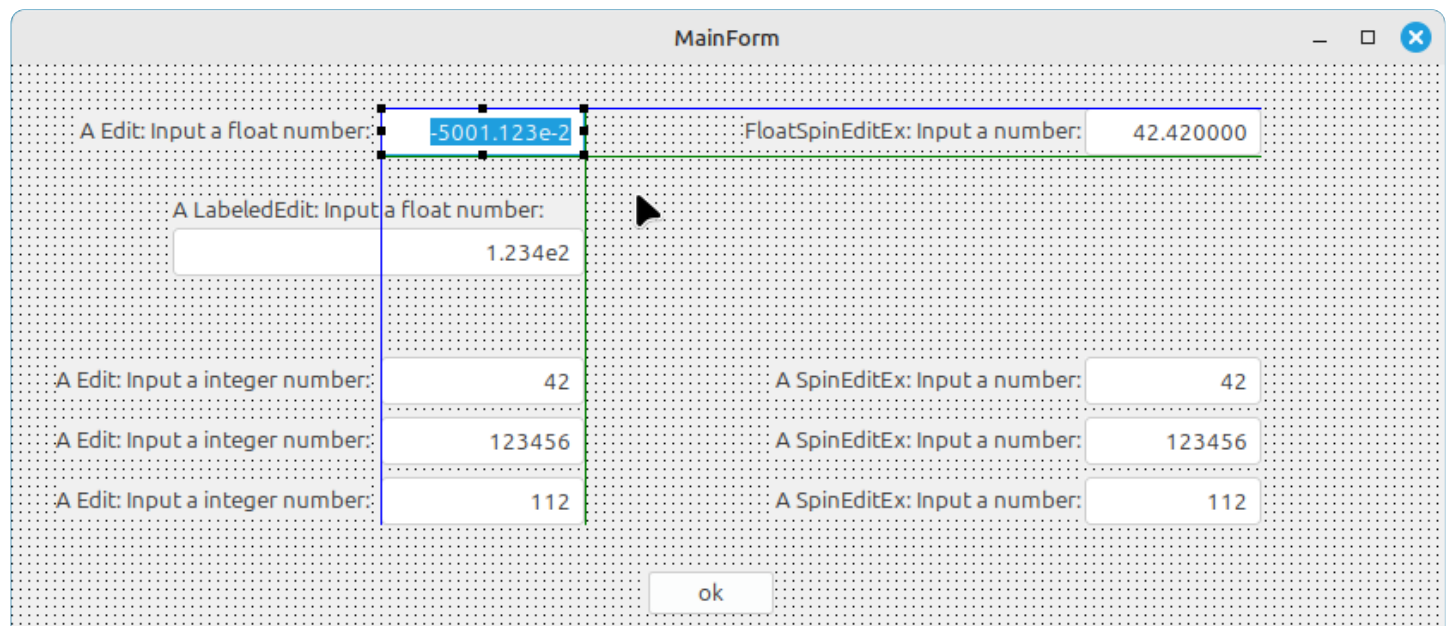


Bild 4: Das MainForm im Formulareditor

Bild 5: Im Objectinspektor das Ereignis "OnKeyPress" mit der procedure "FloatInput" verknüpfen

Flowchart procedure MyFloatInput

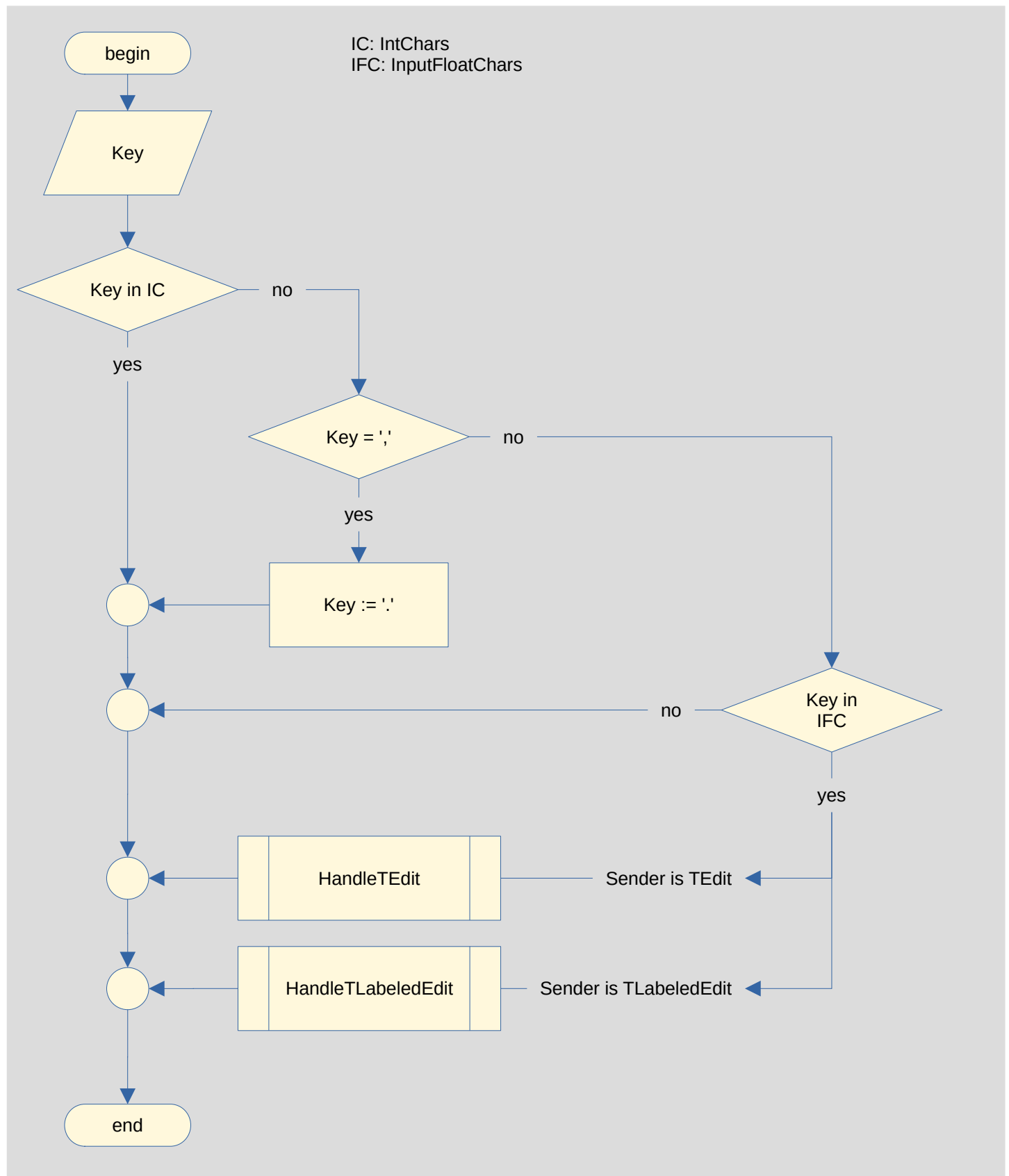


Bild 6: Flowchart zu procedure *MyFloatInput*

Flowchart Procedure HandleTEdit

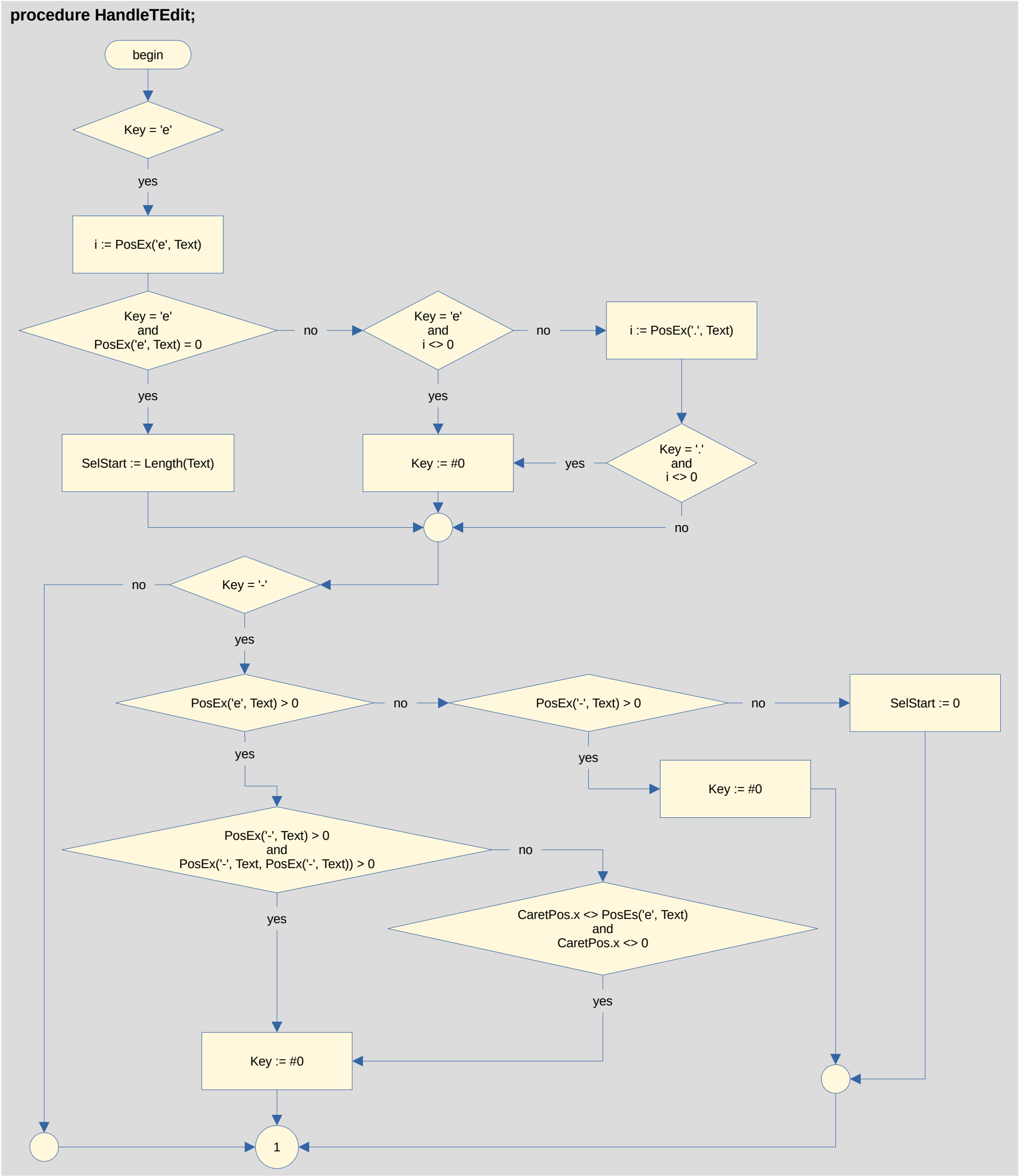


Bild 7: Flowchart zur procedure HandleTEdit

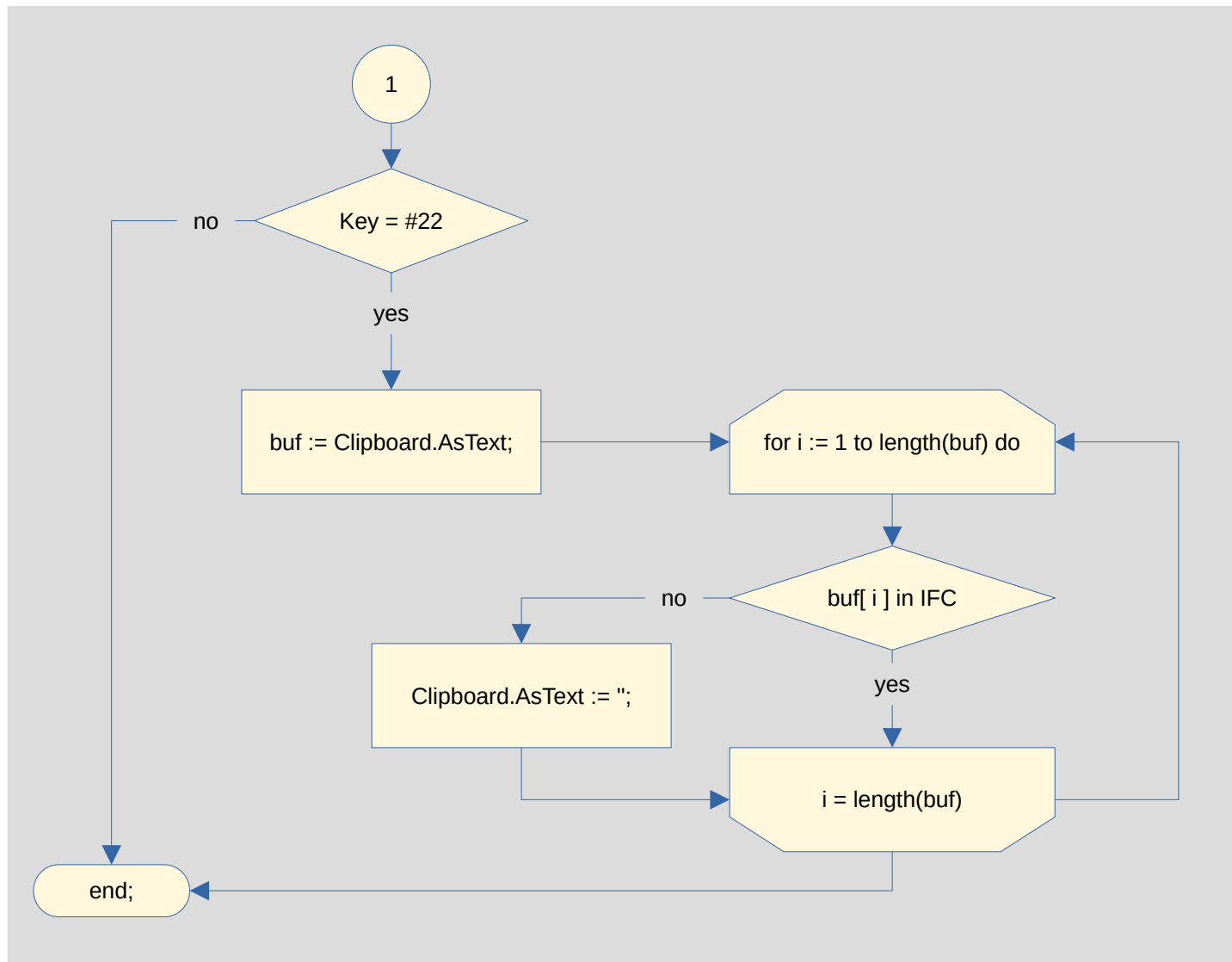


Bild 8: Fortsetzung zur procedure HandleTEdit