FORMS2.INC

Formulare für Turbo Pascal 3 für CP/M

Die Standardeingabeprozeduren von Turbo Pascal via Read / ReadLn genügen für einfache Aufgaben, aber für anspruchsvollere Anwendungen werden fortgeschrittenere Funktionen benötigt. So sollen z.B. bei der Eingabe von ganzen Zahlen nur die Ziffern 0 bis 9 erlaubt sein bei einer frei definierbaren maximalen Eingabelänge. Oder die Eingabe eines Dateinamens soll auf das 8.3-Format beschränkt sein.

FORMS2.INC ist eine Bibliothek zur Definition und zum Betrieb von Eingabeformularen unter Turbo Pascal. Es benötigt die abhängigen Bibliotheken CONVERT.INC, INPUT2.INC und BOX.INC. Die vier Include-Dateien müssen zu Beginn des Programms eingebunden werden:

```
program Anwendung;
(*$I CONVERT.INC*)
(*$I INPUT2.INC*)
(*$I BOX.INC*)
(*$I FORM2.INC*)
```

Ein Formular besteht aus einer doppelt verketteten Liste von Formularelementen. Ein Formularelement ist entweder ein Feld (FEField) oder eine Schaltfläche (FEButton). Eigenschaften der Formularelemente sind u.a. die Positionierung auf dem Bildschirm, der eingegebene Text oder die Aktivierung einer Schaltfläche.

Feld – FEField

Folgende Feldtypen werden unterstützt:

Feldtyp	Beschreibung	Beispieleingabe
FTText	Text mit bis zu 255 Zeichen	Hallo, Welt!
FTNumNatural	Natürliche Zahlen, nur die Ziffern 0 – 9 sind erlaubt	2533
FTInteger	Ganze Zahlen, die Ziffern 0 – 9 und ein Vorzeichen (+ oder -) sind erlaubt	-1398
FTReal	Reelle Zahlen, die Ziffern 0 – 9, ein Dezimalzeichen[1], Vorzeichen (+ oder -) und ggf. ein Exponent sind erlaubt	2,997925E+5
FTDate	Datum im Format TT.MM.JJJJ	18.03.2021
FTFileName	Dateiname im 8.3-Format mit optionaler Laufwerksangabe	M:HALLO.TXT

Felder haben einen Typ (s.o.), eine Länge (fieldLength), eine Beschriftung (caption), die Position der Beschriftung (xCaption, yCaption), die Position des Eingabebereichs (xInput, yInput) und den eingebenen Text (txt). Beschriftung und Text sind dynamische Variablen, ihr Speicherbedarf beträgt Beschriftungs- bzw. Textlänge + 1 Bytes.

1 Das Dezimalzeichen wird über die Prozedur **SetDecimalChar** festgelegt.

Schaltfläche – FEButton

Schaltflächen haben eine Position (x, y), eine Breite (width), eine Beschriftung (caption) und einen Aktivierungszustand (activated). Sie werden mit der Return-Taste aktiviert. Wird eine Schaltfläche mit der Tab-Taste oder mit einer der Pfeiltasten verlassen, dann ist sie nicht aktiviert.

Gebräuchlich sind OK- und Abbrechen-Schaltflächen zum Bestätigen bzw. Stornieren der Formulareingabe.

Initialisierung – Definition der Formularelemente

Ein Design-Ziel der Formularbibliothek war, die Verwendung der Funktionalität möglichst einfach zu gestalten und möglichst viel von den Interna zu verbergen. Die Initialisierung des Formulares erfolgt daher über die Prozedur **InitForm** mit Angabe des Formularindex; für die Definition der Formularelemente muss eine Prozedur namens **InitFormElements** mit dem Formularindex als Parameter geschrieben werden. In letzterer werden die Elemente durch Aufrufe von **AddField** oder **AddButton** definiert und mittels **newAndLinkWithPrevElement** aneinandergehängt. Details folgen in der Beispielanwendung test3 (s.u.).

Speicherverwaltung

Da die Formulardaten mit dynamischen Variablen arbeiten, sollten sie freigegeben werden, wenn sie nicht mehr benötigt werden. Am einfachsten geschieht dies mit Hilfe der Turbo Pascal Prozeduren **Mark** und **Release**; die Variable form_HeapTop befindet sich bereits in der Include-Datei:

```
Mark(form_HeapTop);
…
Release(form_HeapTop);
```

Im Anschluss an den Release-Aufruf sind die Formulardaten nicht mehr verfügbar.

Formular starten

Das Formular wird mit der Prozedur **ProcessForm** gestartet. Der Aufruf geschieht für gewöhnlich direkt im Anschluss an die Initialisierung per InitForm.

Formularbedienung

Wenn das Formular über ProcessForm gestartet wurde, erscheinen die Beschriftungen und Bedienelemente an den angegebenen Positionen auf dem Bildschirm und der Cursor steht im ersten Element und es kann mit der Eingabe begonnen werden. Die Bedienung sollte weitgehend intuitiv sein.

Eingabefeld

Die maximale Länge der Eingabe wird durch eine Linie hinter der Beschriftung angezeigt. Der Cursor steht zu Beginn am Anfang des Feldes und wandert bei Eingaben nach rechts. Er kann innerhalb des Eingabebereichs mit den Tasten Pfeil-Links und Pfeil-Rechts bewegt werden, um an

anderer Stelle Eingaben einzufügen. Mit der Rücktaste und der Entf-Taste können Eingaben an der Cursor-Position nach links bzw. rechts gelöscht werden. Durch Drücken auf die Tab- oder die Pfeil-Unten-Taste wechselt der Cursor ins nächste Feld; mit der Pfeil-Oben-Taste ins vorherige.

Schaltfläche

Auf Schaltflächen ist die einzige Eingabemöglichkeit die Return-Taste zum Aktivieren. Mit den Pfeil-Tasten oder der Tab-Taste wird die Schaltfläche verlassen ohne sie zu aktivieren.

Zugriff auf Formularelemente

Zwei Funktionen ermöglichen, bequem auf bestimmte Formularelemente zuzugreifen: mit **GetFormElement** erhält man das n-te Element in der Liste, mit **GetButton** die n-te Schaltfläche (wobei hier nur die Schaltflächen gezählt werden).

So bekommt man z.B. mit `element := GetFormElement(5)` einen Zeiger auf das fünfte Formularelement als `FormElementPtr`. Oder mit `button := GetButton(1)` die erste Schaltfläche im Formular als `ButtonPtr`. Anschließend kann man über die Pointer auf die Eigenschaften der Elemente zugreifen.

Beispiel GetFormElement

```
var elem:  FormElementPtr;
    field: FieldPtr;
elem  := GetFormElement(3); (* ein Textfeld, d.h. elementType = FTText *)
field := elem^.field;
WriteLn("Feldinhalt: ", field^.txt);
```

Beispiel GetButton

```
var button: ButtonPtr;
button := GetButton(2);
Write("Button aktiviert: ");
if (button^.activated) then
  WriteLn("ja");
else
  WriteLn("nein");
```

Datenkonvertierung

Formularfelder speichern die eingegebenen Werte immer als Text. Hat man bespielsweise einen Wert in ein Ganzzahlfeld eingegeben, dann ist er im Formularfeld immer noch als Zeichenfolge abgelegt, nicht als Integer. Die Konvertierung wird durch Funktionen in CONVERT.INC unterstützt:

Funktion	Beschreibung
CharPtrToString	Formularfeld-Text (txt) in String
StringToCharPtr	String in Formularfeld-Text (txt)
StringToInteger	String in Integer; bei Fehler wird 0 zurückgeliefert

StringToReal	String in Real; bei Fehler wird 0 zurückgeliefert
CharPtrToInteger	Formularfeld-Text (txt) in Integer; bei Fehler wird 0 zurückgeliefert
CharPtrToReal	Formularfeld-Text (txt) in Real; bei Fehler wird 0 zurückgeliefert

Wenn ein Fehler bei der Konvertierung aufgetreten ist, dann ist die Variable **cvt_error** true, ansonsten ist sie false.

WriteCharPtrString

Weil insbes. die txt-Eigenschaft wegen des dynamischen Charakters kein String ist, sondern ein Zeiger auf Character, kann der Text nicht einfach mit `Write(field^.txt)` ausgegeben werden. Stattdessen verwendet man `WriteCharPtrString(field^.txt)`.

Beispielanwendung test3

Die Anwendung test3 erzeugt nacheinander zwei Formulare und startet sie. Nach Beendigung der Eingabe mit der OK- oder der Abbrechen-Schaltfläche werden jeweils die Feldinhalte und der Aktivierungszustand der Schaltflächen angezeigt.

```
program test3;
(*$I CONVERT.INC*)
(*$I INPUT2.INC*)
(*$I BOX.INC*)
(*$I FORM2.INC*)

var
  element: FormElementPtr;

(* Initializes the form elements. This procedure is called from InitForm. *)
procedure InitFormElements;
begin
  case formIdx of
    1: begin
          addField(FTText, 20, 'Name', 10, 6, 17, 6);

          newAndLinkWithPrevElement;
          addField(FTInteger, 5, 'Int', 10, 8, 17, 8);

          newAndLinkWithPrevElement;
          addField(FTNumNatural, 3, 'Gr|~e in cm', 10, 10, 24, 10);

          newAndLinkWithPrevElement;
          addField(FTReal, 10, 'Real', 10, 12, 24, 12);

          newAndLinkWithPrevElement;
          addField(FTDate, 10, 'Geburtsdatum', 10, 14, 24, 14);

          newAndLinkWithPrevElement;
          addField(FTFileName, 14, 'Dateiname', 10, 16, 24, 16);

          newAndLinkWithPrevElement;
          addButton(1, 18, 15, 'OK');
```

```
          newAndLinkWithPrevElement;
          addButton(20, 18, 15, 'Abbrechen');
        end;
     2: begin
          addField(FTDate, 10, 'Datum', 15, 6, 25, 6);

          newAndLinkWithPrevElement;
          addButton(15, 9, 15, 'OK');

          newAndLinkWithPrevElement;
          addButton(33, 9, 15, 'Abbrechen');
        end;
   end; (* case *)
end; (* InitFormElements *)


(* Writes the current field's text and moves to the next element. *)
procedure NextField;
begin
  WriteCharPtrString(element^.field^.txt);
  element := element^.next;
  WriteLn;
end; (* NextField *)


(* Writes the current button's activation status and moves to the next element.
*)
procedure NextButton;
begin
  if element^.button^.activated then WriteLn('yes') else WriteLn('no');
  element := element^.next;
end; (* NextButton *)


(* Displays a headline. *)
procedure Headline;
begin
  ClrScr;
  WriteLn('FORM2-Test');
  WriteLn; WriteLn;
end; (* Headline *)

begin
  SetDecimalChar(',');

  Headline;
  Write('*** Form 1 ***');

  Mark(form_HeapTop);
  InitForm(1);
  ProcessForm;

  GotoXY(1, 22);
  element := form_FirstElement;
  Write('Name        : '); NextField;
  Write('Int         : '); NextField;
  Write('Gr|~e       : '); NextField;
  Write('Real        : '); NextField;
  Write('Geburtsdatum: '); NextField;
  Write('Dateiname   : '); NextField;
  Write('OK          : '); NextButton;
  Write('Abbrechen   : '); NextButton;

  Release(form_HeapTop);
```

```
  WaitForKey;

  Headline;
  Write('*** Form 2 ***');

  Mark(form_HeapTop);
  InitForm(2);
  ProcessForm;

  GotoXY(1, 22);
  element := form_FirstElement;
  Write('Datum       : '); NextField;
  Write('OK          : '); NextButton;
  Write('Abbrechen   : '); NextButton;

  Release(form_HeapTop);
  WaitForKey;
end.
```