Daniel Seefeld 28. 8. 95
Jan Krüger

Programm ASSM03

Bewertung

Programmfunktion: - Pentium fehlt am Ende (vergessen?)
- ausgereift 100%
- einfache Algorithmen
- Stack verwendet

Doku - Pap: Labels (gleiche wie im Programm) machen übersichtlich
- Speicher u. Stackskizze fehlt
- Programmkommentierung ausführlich und funktionell

↳ 14 P Weiter so!

Le

```
unit Kunde;
{
Autor: Jan H. Krüger
Klasse: 12 BG
Tutor: Herr Ruddat
Kursleiter: Herr Brauburger
Name des Programmes: Kundenverwaltung
Version: 1.0.0.17
Funktion: Dieses Programm soll anhand einer einfachen Kundenverwaltung der Umgang mit
 Dateien verdeutlichen. Elemente davon: Laden und Speichern einer Datei, Ändern vorhandener
 Dateisätze, Hinzufügen neuer Datensätze
}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, UElement, Menus, ExtCtrls;

type
  TForm1 = class(TForm)
    sbtnprev: TSpeedButton;
    sbtnNext: TSpeedButton;
    Label1: TLabel;
    edtAusgabe: TEdit;
    btnSpeichern: TButton;
    btnLaden: TButton;
    dlgOpen: TOpenDialog;
    MainMenu1: TMainMenu;
    Datei1: TMenuItem;
    Beenden1: TMenuItem;
    edtAnzahl: TEdit;
    Label2: TLabel;
    edtAktuell: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    edtsizeeinzel: TEdit;
    edtsizegesamt: TEdit;
    Label6: TLabel;
    Label7: TLabel;
    edtposprev: TEdit;
    edtposnext: TEdit;
    btnNeuDaten: TButton;
    edtgoto: TEdit;
    btngoto: TButton;
    mmoAlleDatensaetze: TMemo;
    edtalleausgeben: TButton;
    btnLeeren: TButton;
    Panel1: TPanel;
    Panel2: TPanel;
    procedure btnSpeichernClick(Sender: TObject);
    procedure btnLadenClick(Sender: TObject);
    procedure sbtnNextClick(Sender: TObject);
    procedure sbtnprevClick(Sender: TObject);
    procedure Beenden1Click(Sender: TObject);
    procedure btnNeuDatenClick(Sender: TObject);
    procedure btngotoClick(Sender: TObject);
    procedure LadeDatensatz;
    procedure edtalleausgebenClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnLeerenClick(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var Form1: TForm1;
    //Stelle ist eine Merkvariable mit der festgestellt wird auf welchen Datensatz der
 Dateizeiger gerade zeigt. Groesse gibt die Gesamtgröße der Datei an
    Stelle, groesse : integer;
implementation
```

Das Arbeiten mit 'stelle' u. 'groesse' ist für dieses einführende Beispiel sinnvoll. In der Folge können sie darauf verzichten, da die Klasse TFilestream ja mit f.position u. f.size diese Größen immer mitführt. Stelle = 0 für den ersten Datensatz

```
{$R *.DFM}

{-------------------------------------------------------------------------------}

procedure TForm1.LadeDatensatz;
//Funktion:  Hiermit wird ein Datensatz aus der Datei eingelesen und ausgegeben. Ebenfalls
 werden Korrekturen an der VOR- und NACH- Bytepositionen usw. vorgenommen

begin
   //Sezte den Dateizeiger auf den gewünschten Datensatz und lese ihn in die Variable
 'Kundedat', gebe ihn anschliesen aus
   f.seek ((Stelle) * sizeof(TKunde), soFromBeginning);
   f.read(Kundedat, sizeof(TKunde));
   edtAusgabe.text := Kundedat;

   //Ermittlung der Bytepositionen der Datensätze VOR und NACH des aktuellen Datensatzes
   edtposnext.text := inttostr(f.position);
   f.seek ((Stelle-1) * sizeof(TKunde), soFromBeginning);
   edtposprev.text := inttostr(f.position);

   //Stelle sicher das der Dateizeiger für das weitere Vorgehen auf dem aktuell angezeigem
 Datensatz steht und gibt an welche welcher Datensatz gerade angezeigt wird.
   f.seek ((Stelle) * sizeof(TKunde), soFromBeginning);
   edtAktuell.text := inttostr(Stelle+1);
end;

{-------------------------------------------------------------------------------}

procedure TForm1.btnSpeichernClick(Sender: TObject);
//Funktion: Hiermit können Änderungen im gerade geladenem und angezeigtem Datensatz
 gespeichert werden.
begin
  Kundedat := (edtAusgabe.text);
  f.write (Kundedat, sizeof (TKunde));
end;

{-------------------------------------------------------------------------------}

procedure TForm1.btnLadenClick(Sender: TObject);
//Funtion:  Hiermit wird eine Datei geladen

begin
    if dlgOpen.execute then Kundefilename := dlgOpen.filename;
    //Datei wird zum Lesen und Sschreiben geöffnet und danach der erste Datensatz gleich
 gelesen und ausgegeben
    f := TFilestream.Create (Kundefilename, fmOpenReadWrite);
    f.read(Kundedat, sizeof(TKunde));
    edtAusgabe.text := Kundedat;

    //der Zeiger steht auf dem ersten Datensatz
    Stelle := 0;

    //Ermittlung der Gesamtgröße der Datei und anschließende Ausgabe
    groesse := ((f.size) div (sizeof(TKunde)));
    edtAnzahl.text := inttostr(groesse);


    edtAktuell.text := inttostr(Stelle+1);

    //Berechnung und Ausgabe wie groß alle Datensätze zusammen sind und wie groß ein
 Einzelner ist
    edtsizegesamt.text := inttostr (f.size);
    edtsizeeinzel.text := inttostr (sizeof(TKunde));

    //Ermittlung welche Position VOR und NACH dem ersten aktuellen Datensatz vorliegt
    edtposprev.text := '0';
    edtposnext.text := inttostr(f.position);

    //Sicherstellung das der Dateizeiger auf dem aktuell angezeigtem Datensatz steht.
    f.seek ((Stelle) * sizeof(TKunde), soFromBeginning);
end;

{-------------------------------------------------------------------------------}
```

```
procedure TForm1.sbtnNextClick(Sender: TObject);
//Funkion:  Der nächste Datensatz wird geladen

begin
  //Wenn Datensatzzeiger niedriger ist als Gesamtanzahl der Datensatze dann setze den
 Datensatzzeiger auf den nächsten Datensatz
  if Stelle < groesse-1
    then  inc (Stelle)

    //Ansonsten ist Datensatzzeiger gleich der Anzahl an gesamten Datensätzen innerhalb der
 Datei
    else Stelle := groesse-1;

    //Lade den Datensatz und setze die Variablen
    LadeDatensatz;
end;

{-------------------------------------------------------------------------------}

procedure TForm1.sbtnprevClick(Sender: TObject);
//Funktion:  Der vorherige Datensatz wird geladen

begin
  //Wenn Datensatzzeiger größer als Null ist dann wird er auf den vorherigen Datensatz
 gesetzt
  if Stelle > 0
    then dec (Stelle)

    //Ansonsten wird er auf den allerersten Datensatz gesetzt
    else Stelle := 0;

    //Lade den Datensatz und setze die Variablen
    LadeDatensatz;
end;

{-------------------------------------------------------------------------------}

procedure TForm1.Beenden1Click(Sender: TObject);
//Funktion:  Schliesse die Datei und Beende das Programm
begin
  f.Free;
  close;
end;

{-------------------------------------------------------------------------------}

procedure TForm1.btnNeuDatenClick(Sender: TObject);
//Funktion:  Ein Neuer Datensatz wird der Datei hinzugefügt

begin
  //Setze den Dateizeiger auf den allerletzten Datensatz und lese ihn aus um auf einen neuen
 Datensatz zu setzen
  f.seek (f.size, soFromBeginning);
  f.read(Kundedat, sizeof(TKunde));

  //Eingabe in Variable schreiben und in die Datei schreiben
  Kundedat := edtausgabe.text;
  f.write (Kundedat, sizeof(TKunde));

  //Korrektur der Anzeige der Dateigröße, Gesamtanzahl der Datensätze, des aktuellen
 Datensatzen, der Gesamtgröße der Datei sowie der Bytepositionen vor und nach des aktuellen
 Datensatzes. Aktueller Datensatz ist der letzte, gerade hinzugefügte Datensatz
  groesse := ((f.size) div (sizeof(TKunde)));
  edtAnzahl.text := inttostr(groesse);
  edtAktuell.text := inttostr(Stelle+1);
  edtsizegesamt.text := inttostr (f.size);
  Stelle := groesse-1;
  edtAktuell.text := inttostr(Stelle+1);
  edtposnext.text := inttostr(f.position);
  f.seek ((Stelle-1) * sizeof(TKunde), soFromBeginning);
  edtposprev.text := inttostr(f.position);
end;
```

```
{-------------------------------------------------------------------------------}

procedure TForm1.btngotoClick(Sender: TObject);
//Funktion:  Springt zu einem beliebigem Datensatz innerhalb der Datei und gibt ihn aus.
begin
  Stelle := (strtoint(edtgoto.text)-1);
  LadeDatensatz;
end;

{-------------------------------------------------------------------------------}

procedure TForm1.edtalleausgebenClick(Sender: TObject);
//Funktion:  Liest alle Datensätze aus und schreibt sie in ein Memofeld
begin
  //Setzt den Dateizeiger auf den ersten Datensatz
  f.seek (0 * sizeof(TKunde), soFromBeginning);

  //Solange der Dateizeiger noch unter der Gesamtgröße der Datei ist wird ein Datensatz
 eingelesen und dem Memofeld hinzugefügt
  while (f.position < f.Size) do begin
    f.read(Kundedat, sizeof(TKunde));
    mmoAlleDatensaetze.Lines.Add (Kundedat);
  end;
end;

{-------------------------------------------------------------------------------}

procedure TForm1.FormCreate(Sender: TObject);
//Funktion: Initialisierung
begin
  Kundefilename := 'Kunde.dat';
  Stelle             := 0;
  edtgoto.text       := '';
  edtAusgabe.text    := '';
  edtAnzahl.text     := '';
  edtAktuell.text    := '';
  edtsizeeinzel.text := '';
  edtsizegesamt.text := '';
  edtposprev.text    := '';
  edtposnext.text    := '';
  edtgoto.text       := '';
end;

{-------------------------------------------------------------------------------}

procedure TForm1.btnLeerenClick(Sender: TObject);
//Hiermit kann der Inhalt von bestimmten Editfelder gelöscht werden um die Eingabe eines
 neuen Datensatzes vorzubereiten
begin
  edtAusgabe.text := '';
  edtAktuell.text := '';
  edtposprev.text := '';
  edtposnext.text := '';
  edtgoto.text    := '';
end;

end.
```

```
unit UElement;
{
Autor: Jan H. Krüger
Klasse: 12 BG
Tutor: Herr Ruddat
Kursleiter: Herr Brauburger
Name des Programmes: Kundenverwaltung
Version: 1.0.0.17
Funktion: Dieses Programm soll anhand einer einfachen Kundenverwaltung der Umgang mit
 Dateien verdeutlichen. Elemente davon: Laden und Speichern einer Datei, Ändern vorhandener
 Dateisätze, Hinzufügen neuer Datensätze
}

//Funktion dieser Unit: Bereitstellung des Elementes für das Kundenprogramm

interface

uses classes;

type  TKunde = string [50];

var F : TFilestream;
    Kundefilename : string[50];
    Kundedat : TKunde;
implementation

end.
```