DELPHI 03

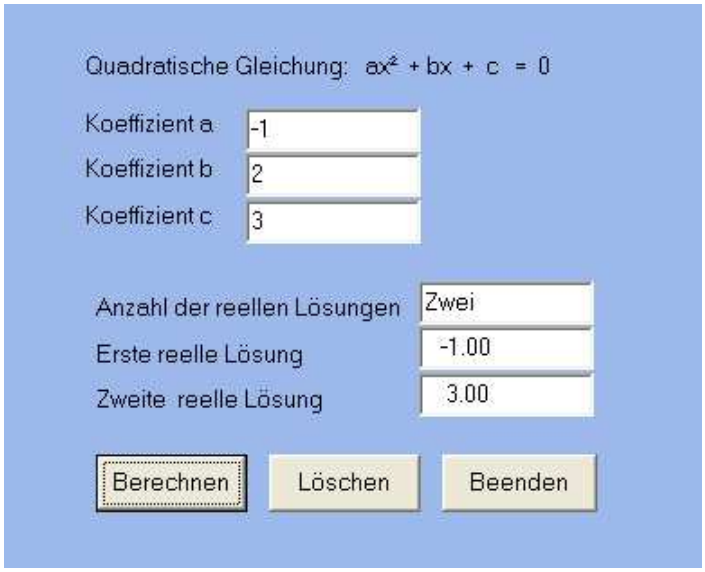
SIEBEN ÜBUNGSPROGRAMME

© Herbert Paukert

[3] SIEBEN EINFACHE ÜBUNGSPROGRAMME

[3.01] Quadratische Gleichungen "*quagl*"

Das Programm soll die reellen Lösungen einer quadratischen Gleichung $ax^2 + bx + c = 0$ ermitteln. Zunächst müssen die drei Koeffizienten a, b und c über entsprechende Editierfelder eingegeben werden. Nach einem Mausklick auf den Schaltknopf ***Berechnen*** erfolgt die Berechnung. Als Ergebnis werden die Anzahl der reellen Lösungen und deren Zahlenwerte in entsprechenden Feldern am Bildschirm ausgegeben. Der Schaltknopf ***Löschen*** löscht alle Datenfelder. Ein Mausklick auf den Schaltknopf ***Beenden*** beendet das Programm.



Wie aus dem Mathematikunterricht bekannt, hängen die Lösungen einer quadratischen Gleichung von den Koeffizienten ab und können auch mit Hilfe einer Formel aus diesen berechnet werden. Die allgemeine Lösungsformel lautet:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Vom Vorzeichen des Ausdruckes unter der Quadratwurzel hängt es ab, ob es reelle Lösungen gibt oder nicht. Dieser wichtige Wert wird auch Diskriminante ***disk*** genannt ($disk = b^2 - 4ac$). Ist sie negativ, dann darf keine Wurzel berechnet werden, weil sonst ein Fehlerabbruch (Exception) des Programmes erfolgen würde. Außerdem müssen die in den Textfeldern eingegebenen Koeffizienten in Zahlen umgewandelt werden. Dazu wird anstelle von ***StrToFloat*** die Umwandlungsprozedur ***Val*** verwendet. Wenn unerlaubte Textzeichen eingetastet worden sind, dann liefert ***Val*** einen Fehlerparameter ***Code*** zurück, der ungleich Null ist. Zur Zahlenausgabe wird anstelle von ***FloatToStr*** die Prozedur ***Str*** verwendet.

Nur wenn alle drei Koeffizienten echte Zahlenwerte sind und nur wenn die Diskriminante nicht negativ ist, kann die Berechnung der Lösungen entsprechend obiger Formel durchgeführt werden. Ist die Diskriminante positiv, dann gibt es immer zwei verschiedene reelle Lösungen ***x1*** und ***x2***. Ist die Diskriminante hingegen Null, dann gibt es nur eine Lösung ***x1***. Im Programm muss daher zuerst entschieden werden, ob ***disk < 0*** ist. Wenn ja, dann gibt es KEINE reelle Lösung und das Programm wird beendet. Wenn nein, dann werden die Lösungen ***x1*** und ***x2*** entsprechend obiger Formel berechnet und ausgegeben. Hier muss aber zusätzlich noch entschieden werden, ob ***disk > 0*** oder ***disk = 0*** ist. Im einen Fall wird für die Anzahl der Lösungen ZWEI, im anderen Fall jedoch EINS ausgegeben. Die einzelnen Fallunterscheidungen werden programmtechnisch durch zwei geschachtelte "***if – then – else***"–Anweisungen realisiert.

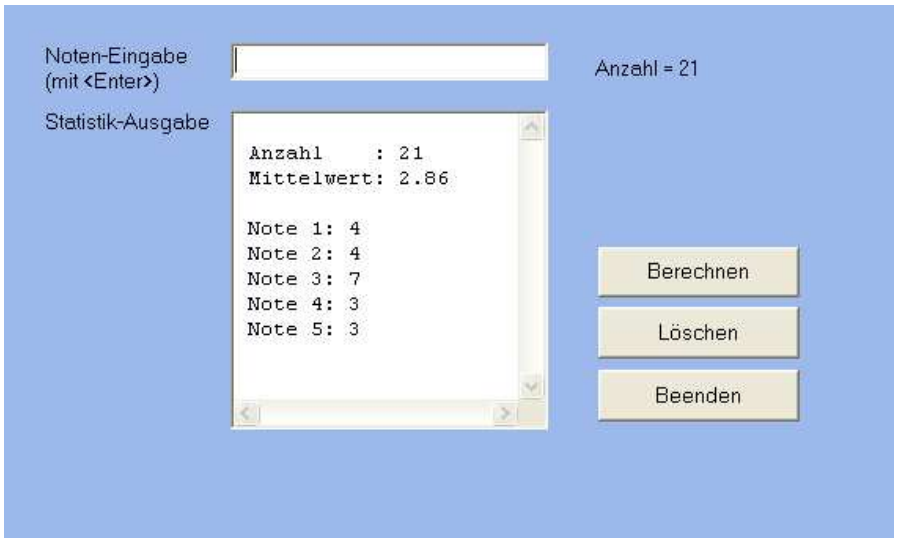
unit quagl_u;

```
// Quadratische Gleichungen (c) H.Paukert

interface

uses Windows, Messages, SysUtils, Classes, Graphics,
     Controls, Forms, Dialogs,  StdCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    private { Private declarations }
    public  { Public declarations }
  end;

var Form1: TForm1;


implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
// Berechnen
var S        : String;
    a,b,c    : Real;
    Code,Err : Integer;
    Disk     : Real;
    x1, x2   : Real;
begin
   Err := 0;

   S := Edit1.Text;
   Val(S,a,Code);
   Err := Err + Code;

   S := Edit2.Text;
   Val(S,b,Code);
   Err := Err + Code;

   S := Edit3.Text;
   Val(S,c,Code);
   Err := Err + Code;

   Disk := b*b - 4*a*c;

   if (Err > 0) or (Disk < 0) then begin
      Edit4.Text := 'Keine';
      Edit5.Text := '';
      Edit6.Text := '';
   end
```

```
   else begin
      Disk := Sqrt(Disk);
      x1   := (-b+Disk)/(2*a);
      x2   := (-b-Disk)/(2*a);
      Str(x1:8:2,S);
      Edit5.Text := S;
      Str(x2:8:2,S);
      Edit6.Text := S;
      if Disk = 0 then Edit4.Text := 'Eine'
                  else Edit4.Text := 'Zwei';
   end;
end;

procedure TForm1.Button2Click(Sender: TObject);
// Löschen
begin
  Edit1.Clear;
  Edit2.Clear;
  Edit3.Clear;
  Edit4.Clear;
  Edit5.Clear;
  Edit6.Clear;
end;

procedure TForm1.Button3Click(Sender: TObject);
// Beenden
begin
  Application.Terminate;
end;

end.
```

[3.02] Die Statistik von Schulnoten "*noten*"

Im Hauptspeicher wird ein Bereich (Array) für hundert ganze Zahlen, eben unsere Schulnoten, reserviert. Während der Eingabe im Editierfeld werden fortlaufend nur entsprechende numerische Werte in das Array abgespeichert, andere Werte hingegen ignoriert. Jede Eingabe muss mit der ***Enter***-Taste abgeschlossen werden, nur dann erfolgt die nächste Zahleneingabe. Gleichzeitig wird die Anzahl der eingetasteten Zahlen ermittelt. Wenn ein Mausklick auf den Schalter ***Berechnen*** erfolgt, dann werden in einer Wiederholungsschleife die Notenhäufigkeiten und die Summe aller Noten berechnet. Danach wird der Mittelwert der Noten berechnet. Die Ausgabe der Anzahl, des Mittelwertes und der Häufigkeiten der Noten erfolgt in einem Memo-Feld. Ein Mausklick auf den Schaltknopf ***Löschen*** löscht alle Datenfelder. Der Schaltknopf ***Beenden*** beendet das Programm.



Zur fortlaufenden Eingabe im Editfeld ***Edit1*** muss im Objektinspektor die Ereignisbehandlungsroutine ***TForm1.Edit1KeyUp*** angeklickt werden. In dieser werden der Tastencode der ***Enter***-Taste abgefragt und dann die entsprechenden Anweisungen zur Zahlenspeicherung durchgeführt.
Erwähnenswert ist, dass die Zeilen (*Lines*) von Memofeldern (*TMemo*) oder die Einträge (*Items*) von Listboxen (*TListbox*) vom Datentyp ***TStringList*** sind. Dabei handelt es sich um eine Klasse von Objekten, welche Listen von Zeichenketten darstellen. Die Eigenschaften und Methoden von ***TStringList*** bieten zahlreiche Möglichkeiten zur Listenverarbeitung (Einfügen und Löschen, Suchen und Sortieren, Speichern in eine Datei und Laden). Die hauptsächlich verwendete Methode *Memo1.Lines.Add(S)* fügt dem Memofeld eine neue Zeile hinzu, in welcher dann der übergebene Stringparameter *S* steht. Die Methode *Memo1.Clear* löscht das ganze Memofeld.

Erfolgt im Objektinspektor ein Doppelklick auf die Lines-Eigenschaft einer Memo-Komponente oder auf die Item-Eigenschaft einer Listbox-Komponente, so erscheint ein StringListen-Editor, der verschiedene Möglichkeiten zur direkten Bearbeitung von Stringlisten bietet. Diese Textzeilen (Lines) erscheinen dann, wenn die entsprechende Komponente aktiviert wird.

Wichtig: Die Anweisung ***Exit*** beendet unbedingt ein laufendes Unterprogramm.

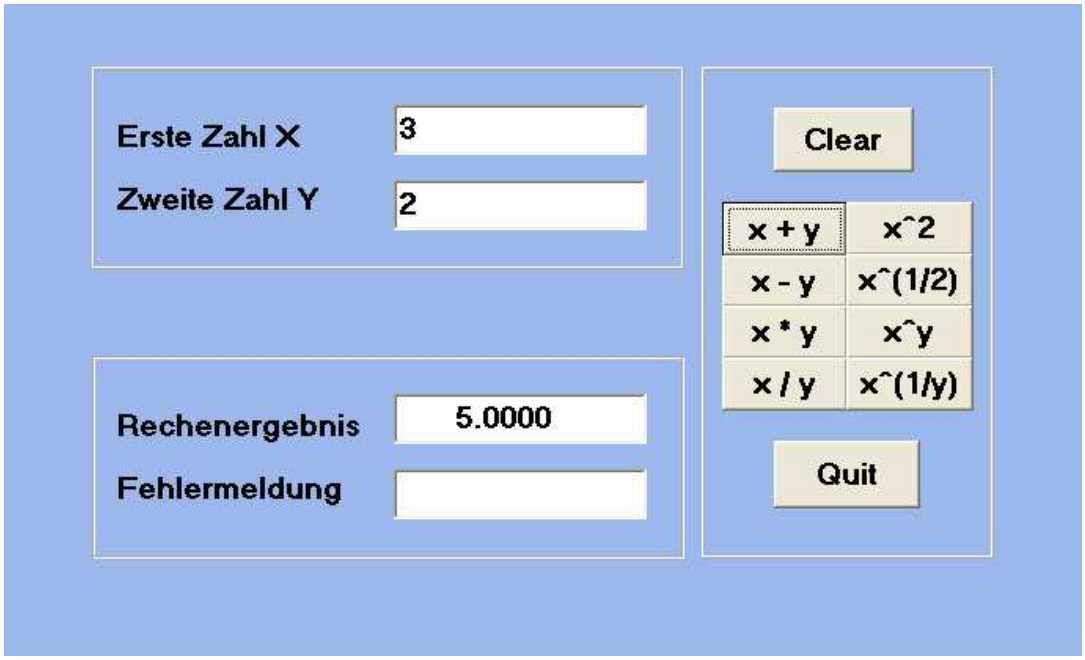
```
unit noten_u;
// Statistische Auswertung von Schulnoten (c) H.Paukert

interface
uses Windows, Messages, SysUtils, Classes, Graphics,
     Controls, Forms, Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
```

```
    Edit1: TEdit;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Label4: TLabel;
    procedure FormActivate(Sender: TObject);
    procedure Edit1KeyUp(Sender: TObject; var Key: Word;
                         Shift: TShiftState);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    private { Private declarations }
    public  { Public declarations }
  end;

var Form1: TForm1;

implementation
{$R *.DFM}

const Max   = 100;
type  Feld  = array[1..Max] of Integer;
      Frequ = array[1..5] of Integer;
var   Note  : Feld;
      NH    : Frequ;
      Anz   : Integer;
      Sum   : Integer;

procedure ClearData;
// Alle Daten löschen
var I : Integer;
begin
  Anz := 0;
  Sum := 0;
  for I := 1 to Max do Note[I] := 0;
  for I := 1 to 5 do NH[I] := 0;
  Form1.Edit1.Clear;
  Form1.Edit1.SetFocus;
end;

procedure TForm1.FormActivate(Sender: TObject);
// Initialisierungen zum Programmstart
begin
  ClearData;
end;

procedure TForm1.Edit1KeyUp(Sender: TObject; var Key: Word;
                            Shift: TShiftState);
// Fortlaufende Dateneingabe mit Gültigkeitsüberprüfung
var  X,Code: Integer;
     S: String;
begin
  if (Edit1.Text = '') or (Anz = Max) then Exit;
  if (Key = 13) then begin
     S := Edit1.Text;
     Val(S,X,Code);
     if (Code > 0) or (X < 1) or (X > 5) then
        ShowMessage('Die Eingabe ist keine Note !')
     else begin
        Anz := Anz + 1;
        Note[Anz] := X;
        Label4.Caption := 'Anzahl = ' + IntToStr(Anz);
     end;
     Edit1.Clear;
     Edit1.SetFocus;
  end;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
// Berechnen und Ausgeben
var S : String;
    X,I : Integer;
    Mwt : Real;
begin
  Sum := 0;
  for I := 1 to Anz do begin
    X := Note[I];
    NH[X] := NH[X] + 1;
    Sum   := Sum + X;
  end;
  if Anz > 0 then Mwt := Sum / Anz
             else Mwt := 0;
  with Memo1 do begin
     Clear;
     Lines.Add(' ');
     S := IntToStr(Anz);  Lines.Add(' Anzahl    : ' + S);
     Str(Mwt:4:2,S);  Lines.Add(' Mittelwert: ' + S);
     Lines.Add(' ' );
     for I := 1 to 5 do begin
         S := ' Note ' + IntTostr(I)+ ': ' + IntToStr(NH[I]);
         Lines.Add(S);
     end;
  end;
  ClearData;
end;

procedure TForm1.Button2Click(Sender: TObject);
// Löschen
begin
  ClearData;
  Memo1.Clear;
  Form1.Label4.Caption := 'Anzahl = 0';
end;

procedure TForm1.Button3Click(Sender: TObject);
// Beenden
begin
  Application.Terminate;
end;

end.
```

[3.03] Ein einfacher Taschenrechner "*rechner*"

Im vorliegenden Programm werden zwei reelle Zahlen in zwei Edit-Komponenten eingegeben. Damit können dann acht verschiedene Rechenoperationen ausgeführt werden. Mit dem Schalter <Clear> wird alles gelöscht, mit <Quit> wird das Programm beendet.




```
unit rechner_u;
//  Unit des Projektes RECHNER (c) H. Paukert.
//  Grundrechenoperatonen mit zwei reellen Zahlen.

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls, ExtCtrls, Math;

type
  TForm1 = class(TForm)
    Label1: TLabel;                  // "Erste Zahl X"
    Label2: TLabel;                  // "Zweite Zahl Y"
    Label3: TLabel;                  // "Rechenergebnis"
    Label4: TLabel;                  // "Fehlermeldung"
    Bevel1: TBevel;                  // erster Rahmen
    Bevel2: TBevel;                  // zweiter Rahmen
    Bevel3: TBevel;                  // dritter Rahmen
    Edit1: TEdit;                    // Eingabe der Zahl X
    Edit2: TEdit;                    // Eingabe der Zahl Y
    Edit3: TEdit;                    // Ausgabe des Ergebnisses
    Edit4: TEdit;                    // Ausgabe der Fehlermeldung
    Button1: TButton;                // <Clear>
    Button2: TButton;                // <Addition>
    Button3: TButton;                // <Subtraktion>
    Button4: TButton;                // <Multiplikation>
    Button5: TButton;                // <Division>
    Button6: TButton;                // <Quadrat>
    Button7: TButton;                // <Quadratwurzel>
    Button8: TButton;                // <Potenz>
    Button9: TButton;                // <Wurzel>
    Button10: TButton;               // <Quit>
```

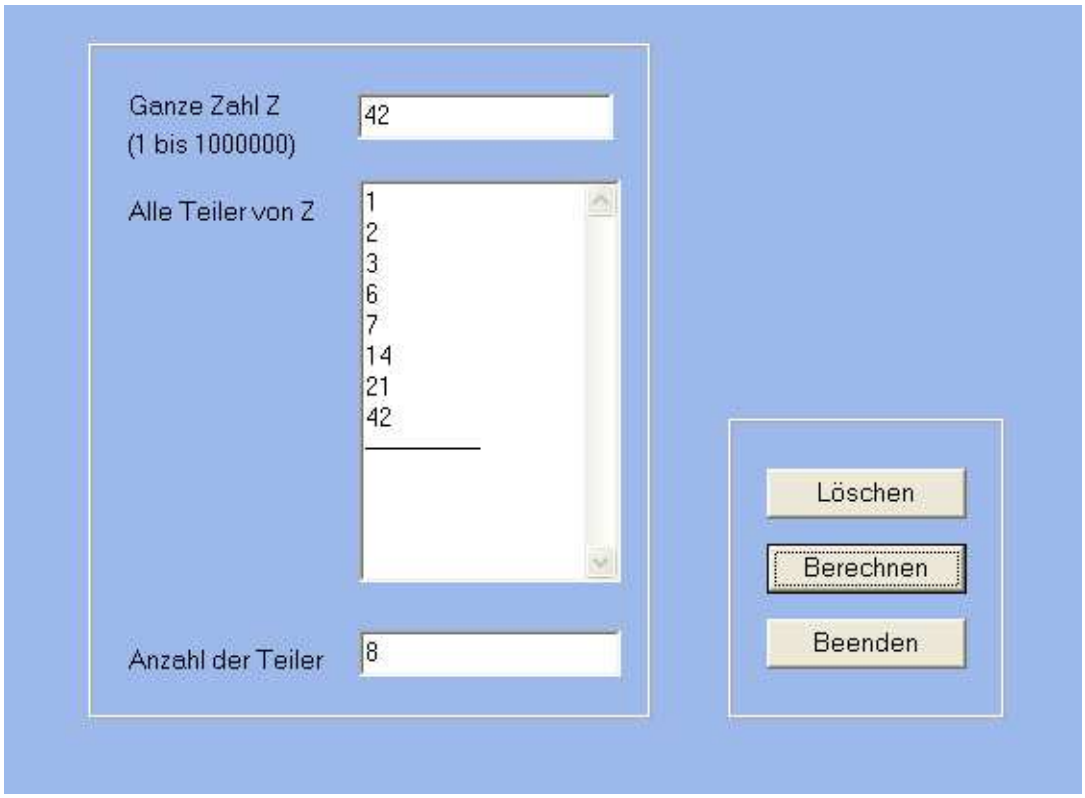
```
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure Button10Click(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
end;

var
    Form1: TForm1;

implementation
{$R *.DFM}
                                    // globale Variable:
var X : Real;                       // Erste Eingabezahl X
    Y : Real;                       // Zweite Eingabezahl Y
    Z : Real;                       // Rechenergebnis
    Error : Boolean;                // Fehlervariable

procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
{ Dateneingaben mit <ENTER> in Edit1 und Edit2 abschließen. }
{ Die KeyPreview-Eigenschaft von Form1 muss True sein.      }
{ Die OnKeyPress-Eventhandler von Edit1 und Edit2 müssen    }
{ auf diese Routine von Form1 gesetzt werden !              }
begin
  if (Sender = Edit1) and (Key = Chr(13)) then
     Edit2.SetFocus;
  if (Sender = Edit2) and (Key = Chr(13)) then
     Edit1.SetFocus;
end;


procedure WertEingabe;
{ Routine zur Zahleneingabe mit Fehlererkennung }
var Code1, Code2: Integer;          // lokale Variable
    S : String;
begin
  S := Form1.Edit1.Text;
  Val(S,X,Code1);                   // Umwandlung von String in Zahl
  S := Form1.Edit2.Text;
  Val(S,Y,Code2);                   // Umwandlung von String in Zahl
  if (Code1 > 0) or (Code2 > 0) then
     Error := True
  else
     Error := False;
end;

procedure WertAusgabe;
{Routine zur Zahlenausgabe mit Fehlermeldung }
var S : String;                     // lokale Variable
begin
   if Not Error then begin
      Str(Z:12:4,S);                // Umwandlung von Zahl in String
      Form1.Edit3.Text := S;
      Form1.Edit4.Text := '';
   end
   else begin
      Form1.Edit3.Text := '';
      Form1.Edit4.Text := ' - Fehler - ';
   end;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
{ Routine zur Löschung der Editierfelder }
begin
   Edit1.Clear;
   Edit2.Clear;
   Edit3.Clear;
   Edit4.Clear;
   Edit1.SetFocus;
end;

procedure TForm1.Button2Click(Sender: TObject);
{ Addition X + Y }
begin
  WertEingabe;
  if Not Error then Z := X + Y;
  WertAusgabe;
end;

procedure TForm1.Button3Click(Sender: TObject);
{ Subtraktion X - Y }
begin
  WertEingabe;
  if Not Error then Z := X - Y;
  WertAusgabe;
end;

procedure TForm1.Button4Click(Sender: TObject);
{ Multiplikation X * Y }
begin
  WertEingabe;
  if Not Error then Z := X * Y;
  WertAusgabe;
end;

procedure TForm1.Button5Click(Sender: TObject);
{ Division X / Y }
begin
  WertEingabe;
  if (Y = 0) then Error := True;
  if Not Error then Z := X / Y;
  WertAusgabe;
end;

procedure TForm1.Button7Click(Sender: TObject);
{ Quadrat von X }
begin
  WertEingabe;
  if (Y=0) then Error := False;
  if Not Error then Z := Sqr(X);
  WertAusgabe;
end;

procedure TForm1.Button8Click(Sender: TObject);
{ Quadratwurzel von X }
begin
  WertEingabe;
  if (Y=0) then Error := False;
  if (X < 0) then Error := True;
  if Not Error then Z := Sqrt(X);
  WertAusgabe;
end;

procedure TForm1.Button9Click(Sender: TObject);
{ Y-te Potenz von X }
begin
  WertEingabe;
  if (X < 0) and (Abs(Y) < 1) then Error := True;
  if Not Error then Z := Power(X,Y);
  WertAusgabe;
end;
```

```
procedure TForm1.Button10Click(Sender: TObject);
{ Y-te Wurzel von X }
begin
  WertEingabe;
  if (X < 0) or (Y = 0) then Error := True;
  if Not Error then Z := Power(X,(1/Y));
  WertAusgabe;
end;

procedure TForm1.Button6Click(Sender: TObject);
{ Programm beenden }
begin
  Application.Terminate;
end;

end.
```

[3.04] Alle Teiler einer Zahl "*teiler*"

Im vorliegenden Programm werden alle Teiler einer eingegebenen Zahl ermittelt.




```
unit teiler_U;
// Alle Teiler einer Zahl (c) H. Paukert

interface
uses Windows, Messages, SysUtils, Classes,
     Graphics, Controls, Forms, Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1:  TEdit;
    Edit2:  TEdit;
    Memo1:  TMemo;
    Bevel1: TBevel;
    Bevel2: TBevel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Label4: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private { Private declarations }
  public  { Public declarations }
end;

var Form1: TForm1;

implementation
{$R *.DFM}
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Lines.Clear;
  Edit1.Clear;
  Edit2.Clear;
  Edit1.SetFocus;
end;

procedure TForm1.Button2Click(Sender: TObject);
var S : String;
    Z,T,N,Code : Integer;
begin
  S := Edit1.Text;
  Val(S,Z,Code);
  if (Code > 0) or (Z < 1) or (Z > 1000000) then begin
     Memo1.Lines.Add('Fehler');
     Exit;
  end;
  N := 0;
  For T := 1 to Z do begin
      if (Z mod T) = 0 then begin
          N := N + 1;
          Str(T,S); Memo1.Lines.Add(S);
      end;
  end;
  Memo1.Lines.Add('---------------');
  Str(N,S); S := S;
  Edit2.Text := S;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Application.Terminate;
end;

end.
```

[3.05] Pythagoräische Zahlen "*pythag*"

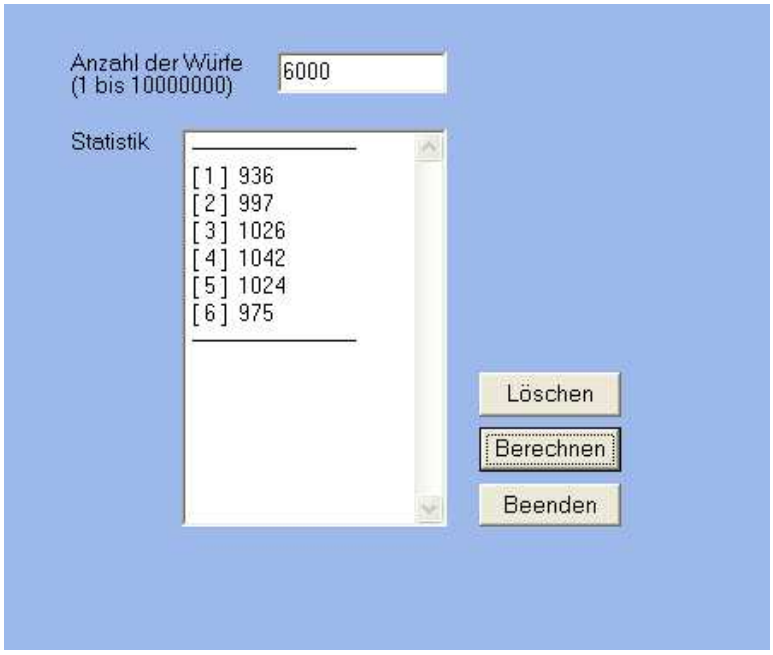
Das vorliegende Programm berechnet schrittweise alle pythagoräischen Zahlentripel (x, y, z) bis zu einer eingegebenen Zahlengrenze N. Dabei gilt $z^2 = x^2 + y^2$.


```
unit pythag_u;
// Pythagoräischen Zahlentripel (c) H. Paukert

interface
uses Windows, Messages, SysUtils, Classes, Graphics,
     Controls, Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1:  TEdit;
    Memo1:  TMemo;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private { Private declarations }
  public  { Public declarations }
end;

var  Form1: TForm1;

implementation
{$R *.DFM}
```

```
procedure TForm1.Button1Click(Sender: TObject);
// Löschen
begin
  Edit1.Clear;
  Memo1.Lines.Clear;
  Edit1.SetFocus;
end;

function GGT(X,Y: Integer): Integer;
// GGT von X und Y
var T, Tmax : Integer;
begin
  Tmax := 1;
  For T := 1 to X do begin
    if ((X mod T) = 0) and ((Y mod T) = 0) then Tmax := T;
  end;
  Result := Tmax;
end;

procedure TForm1.Button2Click(Sender: TObject);
// Berechnen
var Anz,N,X,Y,Z1   : Integer;
    Z          : Real;
    S,SX,Sy,SZ : String;
begin
  Anz := 0;
  N := StrToInt(Edit1.Text);
  if (N < 10) or (N > 1000) then begin
     ShowMessage('Grenze falsch !');
     Edit1.Clear;
     Exit;
  end;
  For X := 1 to N do begin
    For Y := 1 to X do begin
      Z := Sqrt(X*X + Y*Y);
      if (Z < N) and (Frac(Z) = 0) and (GGT(X,Y) = 1) then begin
         Z1 := Round(Z);
         Str(X:8,SX);  SX := Trim(SX);
         Str(Y:8,SY);  SY := Trim(SY);
         Str(Z1:8,SZ); SZ := Trim(SZ);
         S := SX + ', ' + SY + ', ' + SZ;
         Memo1.Lines.Add(S);
         Anz := Anz + 1;
      end;
    end;
  end;
  Str(Anz:8,S); S := Trim(S);
  S := ' Anzahl: ' + S;
  Memo1.Lines.Add(' -------------------- ');
  Memo1.Lines.Add(S);
end;

procedure TForm1.Button3Click(Sender: TObject);
// Beenden
begin
   Application.Terminate;
end;


end.
```

[3.06] Der Zufall würfelt "*zufall*"

In einer Wiederholungsschleife werden die Zahlen 1, 2, 3, 4, 5, 6 zufällig erzeugt. Dann werden die Häufigkeiten der sechs Zufallszahlen ermittelt und ausgegeben.

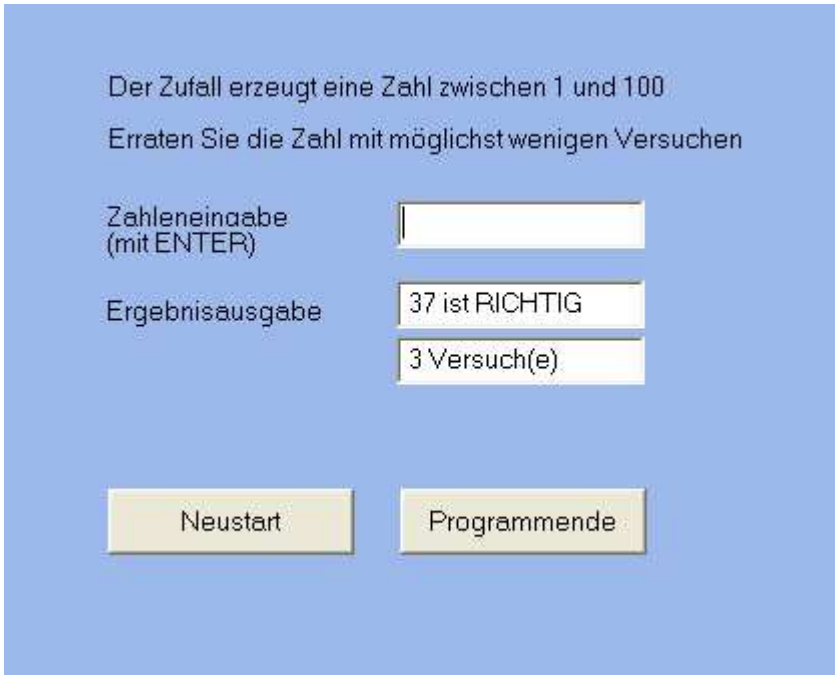



```
unit zufall_u;
// Der Zufall beim Würfeln (c) H. Paukert

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs,StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private { Private declarations }
  public  { Public declarations }
end;

var Form1: TForm1;

implementation
{$R *.DFM}

var  W : Array[1..6] of Integer;   // Ergebniszähler

procedure TForm1.FormCreate(Sender: TObject);
// Zufallsgenerator initialisieren
begin
  Randomize;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
// Ein- und Ausgabefelder löschen
var I : Integer;
begin
  Memo1.Lines.Clear;
  Edit1.Clear;
  Edit1.SetFocus;
  For I := 1 to 6 do W[I] := 0;
end;

procedure TForm1.Button2Click(Sender: TObject);
// Eigentliche Würfel-Routine
var N : Integer;                  // Anzahl der Würfe
    Z : Integer;                  // Zufallsergebnis
    I,Code : Integer;             // Hilfsvariable
    S,T    : String;              // Hilfsvariable
begin
  S := Edit1.Text;
  Val(S,N,Code);
  if (Code > 0) or (N < 1) or (N > 1000000000) then begin
     Memo1.Lines.Add(' FEHLER '); Exit;
  end;
  For I := 1 to N do begin
      Z := Random(6) + 1;
      W[Z] := W[Z] + 1;
//    Str(Z,S); Memo1.Lines.Add(S);
  end;
  Memo1.Lines.Add(' ----------------------- ');
  For I := 1 to 6 do begin
      Str(I,T); T := ' [ ' + T + ' ]  ';
      Str(W[I],S);
      S := T + S;
      Memo1.Lines.Add(S);
  end;
  Memo1.Lines.Add(' ----------------------- ');
end;

procedure TForm1.Button3Click(Sender: TObject);
// Programm beenden
begin
  Application.Terminate;
end;

end.
```

[3.07] Erraten einer Zufallszahl "*raten*"

Zunächst wird eine Zufallszahl zwischen 0 und 101 erzeugt, welche aber nicht sichtbar ist. Diese Zahl muss vom Benutzer erraten werden. Nach jeder Zahleneingabe wird ein Zähler erhöht und als Antwort wird ausgegeben, ob die eingegebene Zahl kleiner oder größer als die Zufallszahl ist. Es werden so lange Zahlen eingegeben bis die Zahl gefunden worden ist.




```
unit raten_U;
// Erraten einer zufälligen Zahl zwischen 1 und 100 (c) H. Paukert

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Button1: TButton;
    Button2: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private  { Private declarations }
  public   { Public declarations }
end;

var  Form1: TForm1;
```

```
implementation
{$R *.DFM}

var X : Integer;                // Zufallszahl
    ANZ : Integer;              // Versuche-Zähler

procedure TForm1.FormCreate(Sender: TObject);
begin
  Randomize;
  X := Random(100) + 1;
  ANZ := 0;
end;

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
var Z : Integer;
    Code : Integer;
    S : String;
begin
  if Key = #13 then begin
     ANZ := ANZ + 1;
     str(ANZ,S);
     Edit3.Text := ' ' + S + ' Versuch(e)';
     S := Edit1.Text;
     Val(S,Z,Code);
     if Z < X then S := ' ' + S + ' ist zu KLEIN';
     if Z > X then S := ' ' + S + ' ist zu GROSS';
     if Z = X then S := ' ' + S + ' ist RICHTIG';
     if Code > 0 then S := ' FEHLER';
     Edit2.Text := S;
     Edit1.Clear;
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ANZ := 0;
  X := Random(100) + 1;
  Edit1.Clear;
  Edit2.Clear;
  Edit3.Clear;
  Edit1.SetFocus;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Application.Terminate;
end;

end.
```