Lösungen der Übungsaufgaben zu zweidimensionalen Feldern

Aufgabe 1: Auslesen von negativen Zahlen aus einer Matrix

```
type
  matrix = array[1..5,1..5] of double;

var
  spalten, zeilen: integer;
  zahlentabelle: matrix;

procedure TForm1.BMatrixErzeugenClick(Sender: TObject);
begin
  if (ESpalte.Text ='') or (EZeile.Text = '') then
    ShowMessage('Bitte geben Sie Zeilen- und Spaltenanzahl ein!')
  else begin
  spalten := StrToInt(ESpalte.Text);
  zeilen := StrToInt(EZeile.Text);
  SGZahlentabelle.ColCount := spalten;
  SGZahlentabelle.RowCount := zeilen;
  SGZahlentabelle.Width := spalten*40;
  SGZahlentabelle.Height := zeilen*24;
  SGZahlentabelle.Visible := True;
  end;
end;

procedure TForm1.BLoeschenClick(Sender: TObject);
var
  z,s: integer;
begin
  ESpalte.Clear;
  EZeile.Clear;
  ENegZahlen.Clear;
  SGZahlentabelle.Visible := False;
  with SGZahlentabelle do for z := 1 to zeilen do
    for s := 1 to spalten do Cells[s-1,z-1] := '';
end;

procedure TForm1.BAuslesenClick(Sender: TObject);
var
 z,s: integer;
 negZahlen: string;
begin
  negZahlen := '';
  with SGZahlentabelle do for z := 1 to RowCount do
    for s := 1 to ColCount do zahlentabelle[s,z] := StrToFloat(Cells[s-1,z-1]);
  for z := 1 to zeilen do
    for s := 1 to spalten do
      if zahlentabelle[s,z] < 0 then negZahlen := negZahlen + ' ' +
FloatToStr(zahlentabelle[s,z]);
  if negZahlen ='' then ENegZahlen.Text := 'Es gibt keine negativen Zahlen!'
  else ENegZahlen.Text := negZahlen;
end;
```

Aufgabe 2: Addition und Subtraktion zweier Matrizen

```
var
  spalten, zeilen: integer;

procedure TForm1.BMatrizenErzeugenClick(Sender: TObject);
begin
  if (ESpalten.Text ='') or (EZeilen.Text = '') then
    ShowMessage('Bitte geben Sie Zeilen- und Spaltenanzahl ein!')
  else begin
  spalten := StrToInt(ESpalten.Text);
  zeilen := StrToInt(EZeilen.Text);
  SGMatrix1.ColCount := spalten;
  SGMatrix1.RowCount := zeilen;
  SGMatrix2.ColCount := spalten;
  SGMatrix2.RowCount := zeilen;
  SGMatrix1.Width := spalten*40;
  SGMatrix1.Height := zeilen*24;
  SGMatrix2.Width := spalten*40;
  SGMatrix2.Height := zeilen*24;
  SGMatrix1.Visible := True;
  SGMAtrix2.Visible := True;
  end;
end;

procedure TForm1.BLoeschenClick(Sender: TObject);
var z,s: integer;
begin
  ESpalten.Clear;
  EZeilen.Clear;
  SGMatrix1.Visible := False;
  SGMatrix2.Visible := False;
  SGAddition.Visible := False;
  SGSubtraktion.Visible := False;
  with SGAddition do for z := 1 to zeilen do
    for s := 1 to spalten do Cells[s-1,z-1] := '';
  with SGSubtraktion do for z := 1 to zeilen do
    for s := 1 to spalten do Cells[s-1,z-1] := '';
  with SGMatrix1 do for z := 1 to zeilen do
    for s := 1 to spalten do Cells[s-1,z-1] := '';
  with SGMatrix2 do for z := 1 to zeilen do
    for s := 1 to spalten do Cells[s-1,z-1] := '';
end;

procedure TForm1.BRechnenClick(Sender: TObject);
type
  matrix = array[1..5,1..5] of double;
var
  matrix1, matrix2, matrixadd, matrixsub: matrix;
  s,z: integer;
begin
  with SGMatrix1 do for z := 1 to RowCount do
    for s := 1 to ColCount do matrix1[s,z] := StrToFloat(Cells[s-1,z-1]);
  with SGMatrix2 do for z := 1 to RowCount do
    for s := 1 to ColCount do matrix2[s,z] := StrToFloat(Cells[s-1,z-1]);
  for z := 1 to zeilen do
    for s := 1 to spalten do begin
      matrixadd[s,z] := matrix1[s,z] + matrix2[s,z];
      matrixsub[s,z] := matrix1[s,z] - matrix2[s,z];
    end;
  SGAddition.ColCount := spalten;
```

```
  SGAddition.RowCount := zeilen;
  SGSubtraktion.ColCount := spalten;
  SGSubtraktion.RowCount := zeilen;
  SGAddition.Width := spalten*40;
  SGAddition.Height := zeilen*24;
  SGSubtraktion.Width := spalten*40;
  SGSubtraktion.Height := zeilen*24;
  SGAddition.Visible := True;
  SGSubtraktion.Visible := True;
  with SGAddition do for z := 1 to RowCount do
    for s := 1 to ColCount do Cells[s-1,z-1] :=
FloatToStrF(matrixadd[s,z],ffFixed,4,2);
  with SGSubtraktion do for z := 1 to RowCount do
    for s := 1 to ColCount do Cells[s-1,z-1] :=
FloatToStrF(matrixsub[s,z],ffFixed,4,2);

end;
```

Aufgabe 3: Magisches Quadrat

```
type
  matrix = array[1..5,1..5] of integer; // Feld für Zahlenquadrat
  reihen = array[1..5] of integer;     // Feld fuer Zeilen- und Spaltensummen

var
  n, dia1, dia2: integer;
  zahlenquadrat: matrix;
  zei, spa: reihen;

procedure TForm1.BLoeschenClick(Sender: TObject);   // Eingaben loeschen
var
  s,z: integer;
begin
  EEingabe.Clear;
  SGZahlenquadrat.Visible := False;
  with SGZahlenquadrat do for z := 1 to n do
    for s := 1 to n do Cells[s-1,z-1] := '';
end;

procedure MatrixEinlesen(var m: Matrix);   // StringGrid in Matrix einlesen
var s,z: integer;
begin
  with Form1.SGZahlenquadrat do for z := 1 to RowCount do
    for s := 1 to ColCount do m[s,z] := StrToInt(Cells[s-1,z-1]);
end;

procedure summen(n: integer; m: matrix; var zsumme,ssumme: reihen; var d1, d2:
integer);
var
  z,s: integer;
begin
  for z := 1 to n do begin     // Ermitteln der Zeilensummen
    zsumme[z] := 0;
    for s := 1 to n do zsumme[z] := zsumme[z] + m[s,z];
  end;
  for s := 1 to n do begin     // Ermitteln der Spaltensummen
    ssumme[s] := 0;
    for z := 1 to n do ssumme[s] := ssumme[s] + m[s,z];
  end;
  d1 := 0;
  d2 := 0;
```

```
  for s := 1 to n do begin  // Diagonalensummen ermitteln
    d1 := d1 + m[s,s];
    d2 := d2 + m[n+1-s,s];
  end;
end;

procedure pruefen(zsumme, ssumme: reihen; n, d1, d2: integer); // Summen pruefen
var
  i: integer;
  wahr: boolean;
begin
  wahr := true;
  if not(d1 = d2) then wahr := false;
  for i := 1 to n do
    if not((zsumme[i] = d1) and (ssumme[i] = d2)) then wahr := false;
  if wahr = true then ShowMessage(' Das Zahlenquadrat ist ein magisches
Quadrat!')
  else ShowMessage('Das Zahlenquadrat ist kein magisches Quadrat!');
end;

procedure TForm1.BPruefenClick(Sender: TObject); // Hauptprogramm
begin
  MatrixEinlesen(Zahlenquadrat);
  summen(n,Zahlenquadrat,zei,spa,dia1,dia2);
  pruefen(zei,spa,n,dia1,dia2);
end;

procedure TForm1.BMatrixErzeugenClick(Sender: TObject); // StringGrid erzeugen
begin
  if Form1.EEingabe.Text ='' then begin
    ShowMessage('Bitte geben Sie die Zeilen- und Spaltenanzahl ein!');
    Form1.EEingabe.SetFocus;
    Exit;
  end;
  n := StrToInt(Form1.EEingabe.Text);
  if (n <= 0) or (n > 5) then begin
    ShowMessage('Bitte geben Sie eine Zahl zwischen 1 und 5 ein!');
    Form1.EEingabe.Clear;
    Form1.EEingabe.SetFocus;
    Exit;
  end
  else begin
    Form1.SGZahlenquadrat.ColCount := n;
    Form1.SGZahlenquadrat.RowCount := n;
    Form1.SGZahlenquadrat.Width := n*40;
    Form1.SGZahlenquadrat.Height := n*24;
    Form1.SGZahlenquadrat.Visible := True;
  end;
end;
```

Aufgabe 4: Verschlüsseln nach der hebräischen Methode

Das Programm funktioniert nur, wenn ein Text mit quadratischer Anzahl von Zeichen eingegeben wir. Bsp.: Regiment marschfähig machen
Ansonsten wird in der letzten Spalte ein Buchstabe eventuell mehrfach ausgelesen!

```
type
  text = array[1..100] of char;
  tabelle = array[1..20,1..20] of char;

var // Globale Variable
  n, laenge: integer;
  klar, geheim: text;
  SG1: tabelle;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Tabsheet1.Show;
end;

{------------------- Löschen der Eintraege -------------------------}

procedure TForm1.BLoeschenClick(Sender: TObject);
var
  s,z: integer;
begin
  Edit1.Clear;
  Edit2.Clear;
  with StringGrid1 do
    for z := 1 to n do
      for s := 1 to n do Cells[s-1,z-1] := '';
  Edit1.SetFocus;
  StringGrid1.Visible := false;
end;

{-------------------------- Loeschen der Eintraege ----------------------}

procedure loeschen;
var
  s,z: integer;
begin
  Form1.Edit1.Clear;
  Form1.Edit2.Clear;
  with Form1.StringGrid1 do
    for z := 1 to n do
      for s := 1 to n do Cells[s-1,z-1] := '';
  Form1.Edit1.SetFocus;
  Form1.StringGrid1.Visible := false;
end;

{---------------- Felder zu Beginn mit Leerzeichen fuellen -----------------}

procedure FelderFuellen(var t1,t2: Text);
var
  i : integer;
begin
  for i := 1 to 100 do begin
    t1[i] := ' ';
    t2[i] := ' ';
  end;
end;

{---------- Leerzeichen aus dem Feld entfernen -------------------------}

function leere_Felder_Test(feld1: text; zeiger: integer): integer;
{Funktion zum Testen der Anzahl der hintereinander(!) liegenden leeren Elemente}
var
  zaehler: integer;
begin
```

```
  zaehler := 0;
  while feld1[zeiger] = ' ' do begin
    inc(zaehler);  {Zähler für die Anzahl der nachzurückenden Elemente}
    inc(zeiger);
  end;
  result := zaehler;
end;

procedure LeerzeichenEntfernen (textlaenge: integer; var t: text);
var
  i, j, AnzLeereFelder: integer;
begin
  for i := 1 to textlaenge do
    if t[i] = ' ' then begin     // pruefen auf Leerzeichen
      AnzLeereFelder := leere_Felder_Test(t,i);
{auf Anzahl der leeren Elemente testen}
      for j := i to textlaenge do t[j] := t[j+AnzLeereFelder]; // Nachruecken
    end;
  laenge := 0;
  repeat  // Laenge des Feldes ohne Leerzeichen ermitteln
    inc(laenge);
  until t[laenge] = ' ';
  laenge := laenge - 1;
end;

{--------------- Feld auf Laenge pruefen ------------}

procedure pruefen (var ok: boolean);
begin
  if laenge > 100 then begin  // pruefen auf Textlaenge
      Showmessage('Der Text ist zu lang!');
      loeschen;
      ok := false;
      exit;
    end;
  ok := true;
end;

{---------------- Matrix erzeugen --------------------------------}

procedure MatrixErzeugen(anz: integer);
begin
  Form1.StringGrid1.ColCount := n;
  Form1.StringGrid1.RowCount := n;
  Form1.StringGrid1.Width := n*20;
  Form1.StringGrid1.Height := n*21;
  Form1.StringGrid1.Visible := True;
end;

{--------------- Klartext verschluesseln --------------------------}

procedure verschluesseln(t: text; var t1: text);
var
  s,z, zaehler: integer;
begin
  zaehler := 1;
  for z := 1 to n do
    for s := 1 to n do begin
      SG1[z,s] := t[zaehler]; // Klartext aus Feld in Tabelle einlesen
      Form1.StringGrid1.Cells[s-1,z-1] := SG1[z,s]; // Ausgabe in StringGrid
      inc (zaehler);
    end;
  zaehler := 1;
```

```
  for s := 1 to n do
    for z := 1 to n do begin
      t1[zaehler] := SG1[z,s];  // Geheimtext aus Tabelle in Feld einlesen
      inc (zaehler);
    end;
end;

{------------------- Geheimtext ausgeben -------------------------------}

procedure AusgabeGeheimtext(t:text);
var
  i: integer;
  ausgabe: string;
begin
  ausgabe := '';
  LeerzeichenEntfernen(laenge,t);
  for i := 1 to laenge do ausgabe := ausgabe + t[i];
  Form1.Edit2.Text := ausgabe;
end;

{------------------ Hauptprogramm Verschluesseln--------------------------}

procedure TForm1.BVerschluesselnClick(Sender: TObject);
var
  klartext: string;
  textlaenge, i: integer;
  pruefe: boolean;
begin
  pruefe := false;
  FelderFuellen(klar,geheim); // Felder mit Leerzeichen initialisieren
  klartext := Edit1.Text;  // Klartext einlesen
  textlaenge := length(klartext); // Laenge des Klartextes ermitteln
  for i := 1 to textlaenge do klar[i] := klartext[i]; // Klartext zeichenweise
in Feld einlesen
  LeerzeichenEntfernen(textlaenge,klar);
  pruefen(pruefe);  // Klartext auf zulaessige Laenge pruefen
  n := trunc(Sqrt(laenge-1))+1; // Anz. der Spalten und Zeilen ermitteln
  if pruefe = true then begin  // wenn Test bestanden
    MatrixErzeugen(n);
    verschluesseln(klar,geheim); // Text verschlüsseln und in StringGrid1
ausgeben
    AusgabeGeheimtext(geheim);  // Geheimtext ausgeben
  end;
end;
```