```
unit unit_BK_Algorithmen;

interface

uses
  db,
  unit_SchildGlob,
  unit_BasisPruefungsalgorithmus;

type
  TBestehungswert = (bwBestanden, bwNichtBestanden, bwIgnorieren );

    TBKPruefungsAlgorithmus = class( TPruefungsAlgorithmus )
        private

      function Teilpruefung: string;override;
            procedure ResetBKMdlPruefungen;
            procedure ResetBKAbschluss( const s_id: integer );
            function Pruefe_APO_BK_15_A_FK_Abschluss: string;// A01-A04
            function Pruefe_APO_BK_15_A_VBG_BSP_Abschluss: string;// A05
            function Pruefe_APO_BK_15_A_VBG_HSA_Abschluss: string;   // Teil von A05

      function Pruefe_APO_BK_15_A12_A13: string;

            function Pruefe_APO_BK_15_A_BG_Abschluss: string;   // A06 nicht mehr vorhanden
            in 15, aber B06, weitgehend identsich

            function Pruefe_APO_BK_15_A_JOA: string;        // A07

            function Pruefe_BK_B_Versetzung: string;
            function BK_B_MuendlichePruefungZulaessig( ds: TDataSet ): boolean;
            function BK_B_KinderpflegerEKurs( const jahr: integer; const msg: boolean ):
            boolean;
            function Pruefe_APO_BK_15_B_BAB_Abschluss_B01_B08_B09_B10: string;  // B01
            function Pruefe_APO_BK_15_B_BAB_Abschluss_B02_B07: string;  // B02
            function Pruefe_APO_BK_15_B_BAB_Abschluss_B03: string;  // B03

            function Pruefe_BK_C_Versetzung: string; // Versetzung C01
      function AbschlussBerechnen_FHR: TBestehungswert;
      function Abschlussberechnen_BA: TBestehungswert;
      function Abschlussberechnen_ErweiterteBeruflicheKenntnisse: TBestehungswert;
      procedure AbschlussErgebnisSpeichern( const bestanden_ab, bestanden_bb:
      TBestehungswert; art_intern_ab, art_intern_bb, statkrz: string;
                                        const meldung_text: string = '' );
      function Zulassung_FHR_BAP( const zulassungsmodus: string ): boolean;
      function Zulassung_ErweiterteBeruflicheKenntnisse: boolean;
            procedure Pruefe_BeruflicheKenntnisse_C03;
      function Pruefe_APO_BK_15_Abschluss_C: string;
      function Pruefe_APO_BK_15_Abschluss_E: string;

            function Pruefe_APO_BK_15_Abschluss_D03_D04: string;

            function Pruefe_APO_BK_15_Abschluss_D05_D06: string;

      procedure BK_B_AbschlussnotenBerechnen( ds: TDataset; const gew_feld: string );
      function NachpruefungsFaecherErmitteln: boolean;

            function BK_NotenDurchschnitt( Tabelle: TDataset; const doppelt: integer; const
            nfld: string ): double;
      function BK_NotenDurchschnittFHR( Tabelle: TDataset; const nfld: string ): double;
      function BK_QVermerkBerechnen: boolean;
            function Pruefe_FP( ds: TDataset; const fld: string; const note: double ): string;
      function BK_B_ZulassungPruefen: boolean;
      function NoteVorhandenGewichtung( Tabelle: TDataset; Feld, Op: string; Note: double;
      const GewichtsFeld: string ): string;
      function IstSpezielleFachklasse( const fkl_liste: string ): boolean;
      function AktuellerAbschlussIstHoeherAlsEingangsqualifikation( const ab_abschl: string
      ): boolean;
```

```
      function POJahrErsetzen( const atxt: string): string;
      function IstBK13: boolean;
      function IstBK11: boolean;
      function IstEGliederung: boolean;
    public

  end;


implementation

uses
  SysUtils,
  adodb,
  RBKLists,
  RBKStrings,
  RBKDBUtils,
  Dialogs,
  StrUtils,
  unit_TransactionHandler,
  unit_SchildFunktionen,
  unit_SchildSettings,
  unit_Mengen,
  BetterADODataset;


const
  C_PO_GLIEDERUNG_A = 'APO-BK-03/A-FK;APO-BK-11/A-FK;APO-BK-11/A-FK/BSA;APO-BK-15/A-FK'; //
  A01
  C_PO_GLIEDERUNG_A_AV = 'APO-BK-03/A-VBG;APO-BK-11/A-VBG'; // A05
  C_PO_GLIEDERUNG_A_BG = 'APO-BK-03/A-BG;APO-BK-11/A-BG'; // A06, Kein Eintrag für 15?
  C_PO_GLIEDERUNG_A_JOA = 'APO-BK-03/A-JOA;APO-BK-11/A-JOA'; // A07, Kein Eintrag für 15?
  C_PO_GLIEDERUNG_A12_A13 = 'APO-BK-15/A-AV';
  C_PO_GLIEDERUNG_B = 'APO-BK-03/B;APO-BK-11/B;APO-BK-15/B';
  C_PO_GLIEDERUNG_C_BAB = 'APO-BK-03/C-BFS-BAB;APO-BK-11/C-BFS-BAB;APO-BK-15/C-BFS-BAB';
  C_PO_GLIEDERUNG_C_BK =
  'APO-BK-03/C-BFS-BK;APO-BK-11/C-BFS-BK;APO-BK-13/C-BFS-BK;APO-BK-15/C-BFS-BK';
// Falscher Eintrag in Statkue: APO-BK-15/C-BFS-BK/BK
  C_PO_GLIEDERUNG_C_FOS = 'APO-BK-03/C-FOS;APO-BK-11/C-FOS;APO-BK-15/C-FOS';
  C_PO_GLIEDERUNG_D_BAB = 'APO-BK-03/D-BAB;APO-BK-11/D-BAB;APO-BK-15/D-BAB';
  C_PO_GLIEDERUNG_D_FOS = 'APO-BK-03/D-FOS13;APO-BK-11/D-FOS13;APO-BK-15/D-FOS13';
  C_PO_GLIEDERUNG_E = 'APO-BK-03/E;APO-BK-11/E;APO-BK-15/E';

  C_FG_ABSCHLUSSARBEIT = 1700;

function TBKPruefungsAlgorithmus.IstBK11: boolean;
begin
  Result := AnsiStartsText( 'APO-BK-11', fPruefOrdnung );
end;

function TBKPruefungsAlgorithmus.IstBK13: boolean;
begin
  Result := AnsiStartsText( 'APO-BK-13', fPruefOrdnung );
end;

function TBKPruefungsAlgorithmus.IstEGliederung: boolean;
begin
  Result := FBKGliederung in [ glE01, glE02, glE03, glE04, glE05, glE06, glE07, glE13 ];
end;


function TBKPruefungsAlgorithmus.POJahrErsetzen( const atxt: string): string;
begin
  if IstBK11 then
    Result := StringReplace( atxt, '15', '11', [] )
  else if IstBK13 then
    Result := StringReplace( atxt, '15', '13', [] )
  else
```

```
    Result := atxt;
end;

function TBKPruefungsAlgorithmus.Teilpruefung: string;
var
  bad: boolean;
begin
  with FC0 do
  begin
    First;
    while not Eof do
    begin
      bad := ( FTextNoten.IndexOf( FieldByname( 'NoteKrz' ).AsString ) >= 0 );
      if bad then
      begin
        if FCS.Locate( 'ID', FieldByName( 'ID' ).AsInteger, [] ) then
          FCS.Delete;
        Delete;
      end else
        Next;
    end;
  end;


  try
// Ab hier die Berufsschulprüfungen
//-------------------------------------------------------------------------------------------
-
// Jetzt der Bildungsgang "A01 Berufsschule Teilzeit, normale Fachklasse"  (A01-A04)
    if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_A ) then
    begin
// Hinweis: Bei dieser Prüfungsordnung werden nicht die einzelnen Teilprüfungsordnungen der
Reihe nach bgearbeitet,
// da diese alle gleichwertig sind und das Ergebnis nur von der Eingangsvoraussetzung abhängt.
      Result := Pruefe_APO_BK_15_A_FK_Abschluss;    // Speicherung erfolgt in Routine
      if ( Result = 'A' ) and ( FBKGliederung = glA02 ) then
        Result := Pruefe_APO_BK_15_Abschluss_C;     // Für A02 wir der gleiche FHr-Algorithmus
      FAbbruch := true;     // Damit nicht alle Teilprüfungsordnungen abgeklappert werden

// ------------------------------------------------------------
// Bildungsgang "Vorklasse zum Berufsgrundschuljahr" A05
// Anforderungen des Bildungsganges erfüllt?
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_A_AV ) then
    begin
      Result := Pruefe_APO_BK_15_A_VBG_BSP_Abschluss;
      Fabbruch := true;
// ------------------------------------------------------------
// Bildungsgang "Berufsgrundschule A06"
// Anforderungen des Bildungsganges erfüllt?
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_A_BG ) then
    begin
      Result := Pruefe_APO_BK_15_A_BG_Abschluss;
      Fabbruch := true;
// ------------------------------------------------------------
// Bildungsgang "Jugendliche ohne Ausbildungsverhältnis A07"
// Anforderungen des Bildungsganges erfüllt?
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_A_JOA ) then
    begin
      Result := Pruefe_APO_BK_15_A_JOA;
      Fabbruch := true;
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_A12_A13 ) then
    begin
      Result := Pruefe_APO_BK_15_A12_A13;
      Fabbruch := true;
//-------------------------------------------------------------------------------------------
-
// Bildungsgang B01 BF zweijährig Berufsabschluss
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_B ) then
```

```
    begin
      if FPruefungsart = 'V' then
      begin     // Versetzung
        fTeilPruefOrdKrz := POJahrErsetzen( 'APO-BK-15/B/V11' );
        Result := Pruefe_BK_B_Versetzung;
        if ( Result = 'N' ) and NachpruefungsFaecherErmitteln then
          Result := 'NP';
        VersetzungSpeichern( Result );
        AbschlussSpeichern( 0 );
      end else
      begin // Abschluss
        case FBKGliederung of
        glB01, glB08, glB09, glB10: Result :=
        Pruefe_APO_BK_15_B_BAB_Abschluss_B01_B08_B09_B10;
        glB02, glB07: Result := Pruefe_APO_BK_15_B_BAB_Abschluss_B02_B07;
        glB03: Result := Pruefe_APO_BK_15_B_BAB_Abschluss_B03;
        glB06: Result := Pruefe_APO_BK_15_A_BG_Abschluss;  // Bildungsgang B06, ist
        weitgehend identsich mit alter A06
        end;
        Fabbruch := true;
      end;
//------------------------------------------------------------------------------------------
-
// Bildungsgang C01 und C02
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_C_BAB ) then
    begin
      if FPruefungsart = 'V' then
      begin     // Versetzung
        fTeilPruefOrdKrz := POJahrErsetzen( 'APO-BK-15/C-BFS-BAB/V' );
        Result := Pruefe_BK_C_Versetzung;
        if ( Result = 'N' ) and NachpruefungsFaecherErmitteln then
          Result := 'NP';
        VersetzungSpeichern( Result );
        AbschlussSpeichern( 0 );
      end else if FPruefungsart = 'A' then
      begin
        Result := Pruefe_APO_BK_15_Abschluss_C;
        Fabbruch := true;
      end;
//------------------------------------------------------------------------------------------
-
// Bildungsgang C03 + C04
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_C_BK ) then
    begin
      if FPruefungsart = 'V' then
      begin     // Versetzung
        fTeilPruefOrdKrz := POJahrErsetzen('APO-BK-15/C-BFS-BK/V');
        Result := Pruefe_BK_C_Versetzung;
        if ( Result = 'N' ) and NachpruefungsFaecherErmitteln then
          Result := 'NP';
        VersetzungSpeichern( Result );
// Berufliche Kenntnisse Ende 1. JG ermitteln
        Pruefe_BeruflicheKenntnisse_C03;
        Fabbruch := true;
      end else if FPruefungsart = 'A' then
      begin
        Result := Pruefe_APO_BK_15_Abschluss_C;
        Fabbruch := true;
      end;
//------------------------------------------------------------------------------------------
-
// Bildungsgang C07 + C08 + C05??
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_C_FOS ) then
    begin
      if ( FPruefungsart = 'V' ) then
      begin     // Versetzung
        case FBKGliederung of
        glC07:
```

```
          begin
            fTeilPruefOrdKrz := POJahrErsetzen('APO-BK-15/C-FOS/V12B');
            VersetzungSpeichern( 'V' );     // Versetzung obligatorisch   nur für C07
            AbschlussSpeichern( 0 );
            Result := 'V';
          end;
        glC05:
          begin
            fTeilPruefOrdKrz := POJahrErsetzen('APO-BK-15/C-FOS/VP');
            Result := Pruefe_BK_C_Versetzung;
            if ( Result = 'N' ) and NachpruefungsFaecherErmitteln then
              Result := 'NP';
            VersetzungSpeichern( Result );
            AbschlussSpeichern( 0 );
          end;
        end;
      end else if FPruefungsart = 'A' then
        Result := Pruefe_APO_BK_15_Abschluss_C;
      Fabbruch := true;
//-------------------------------------------------------------------------------------------------
-
// Bildungsgang D03 + D04
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung, C_PO_GLIEDERUNG_D_BAB ) and (
    FBKGliederung in [ glD03, glD04 ] ) then
    begin
      Result := Pruefe_APO_BK_15_Abschluss_D03_D04;
      Fabbruch := true;
//-------------------------------------------------------------------------------------------------
-
// Bildungsgang D05 + D06
    end else if FBK_Aktiv and IstInListe( fPruefOrdnung,  C_PO_GLIEDERUNG_D_FOS ) then
    begin
// NEU 3.7.17: Die Berechnung in D05 und D06 soll eaxkt so sein wie in der Anlage C
      Result := Pruefe_APO_BK_15_Abschluss_C;//Pruefe_APO_BK_15_Abschluss_D05_D06;
      Fabbruch := Result <> 'A';
//                  abbruch := true;
//-------------------------------------------------------------------------------------------------
-
    end else if FBK_Aktiv and IstTeilInListe( fPruefOrdnung, C_PO_GLIEDERUNG_E ) then
    begin
      if FPruefungsart = 'A' then
      begin
        Result := Pruefe_APO_BK_15_Abschluss_E;
        Fabbruch := true;
      end;
//-------------------------------------------------------------------------------------------------
-
// Noch nicht realisiert
    end else
    begin
      Result := '?';
      Meldung( 'Für die Prüfungsordnung des Schülers ist noch kein Algorithmus vorhanden' );
    end;
    finally
        fTeilPruefOrdKrz := '';
        fTeilPruefOrdLang := '';
    end;

end;


procedure TBKPruefungsAlgorithmus.ResetBKAbschluss( const s_id: integer );
var
    qry: TBetterADODataset;
    s: string;
  cmd: string;
begin
    TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET ' +
```

```
                                                        'Zulassung=NULL,' +
                                                        'Bestanden=NULL,' +
                                                        'ZertifikatBK=NULL,'+
                                                        'ZulassungErwBK=NULL,' +
                                                        'BestandenErwBK=NULL,' +
                                                        'ZulassungBA=NULL,' +
                                                        'BestandenBA=NULL' +
                                                        ' WHERE Schueler_ID=%d', [ s_id
                                                        ] ) );
  cmd := Format( 'UPDATE Schueler SET ' +
                 'EntlassArt=NULL, ' +
                 'Durchschnittsnote=NULL, DSN_Text=NULL,' +
                 'DurchschnittsnoteFHR=NULL, DSN_FHR_Text=NULL WHERE ID=%d',
                 [s_id] );
  TransactionHandler.DoExecute( cmd );
// Prüfen, ob in einem Abschnitt das "Zertifikat berufl. Kenntnisse erreicht (nur bei
C03-Gliederungen)
    if FBKGliederung = glC03 then
    begin
        qry := CreateForwardDataset( fConSchild, SchildSettings.DBFormat <> 'MSACCESS' );
        try
//          qry.CommandText := Format( 'SELECT ID FROM SchuelerLernabschnittsdaten WHERE
Schueler_ID=%d AND Abschluss_B=%s', [ s_id, QuotedStr( 'APO-BK-03/C-BFS-BK/BK' ) ] );
            qry.CommandText := Format( 'SELECT ID FROM SchuelerLernabschnittsdaten WHERE
            Schueler_ID=%d AND Abschluss_B=%s',
                                [ s_id, QuotedStr( POJahrErsetzen( 'APO-BK-15/C-BFS-BK/BK' )
                                ) ] );
            qry.Open;
            if qry.IsEmpty then
                s := '-'
            else
                s := '+';
            TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET
            ZertifikatBK=%s WHERE Schueler_ID=%d',
                                                                [ QuotedStr( s ), s_id ]
                                                                ) );
        finally
            FreeAndNil( qry );
        end;
    end;
end;

procedure TBKPruefungsAlgorithmus.ResetBKMdlPruefungen;
begin
    TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKFaecher SET MdlPruefung=%s,
    MdlPruefungFW=%s, NoteMuendlich=NULL, NoteAbschluss=NULL, NotePrfGesamt=NULL,
    NoteAbschlussBA=Null WHERE Schueler_ID=%d',
                                        [ QuotedStr( '-' ), QuotedStr( '-' ), fS_ID ] ) );
end;

function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_A_FK_Abschluss: string;
// A01, Berufsschule Teilzeit, normale Fachklasse
const
    MsgNeg = 'Kein Abschluss:';
var
    msg, idf: string;
    i : integer;
    DSN : Double;
  diff_ber, cmd: string;
  abschl: string;
begin
    Result := 'N';
    msg := '';

// Differenzierungsbereich raus
// Neu 30.1.2011: Diff-Bereich nur temporär entfernen

  diff_ber := FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 );
```

```
  Uebertragen( fC0, fC1, diff_ber );        // von C0 nach C1
//  AusContainerLoeschen( fC0, diff_ber );

  try
  // Anzahl Noten = 6 > 0?
    idf := NoteVorhanden( fC0, 'Note', '=', 6, -1 );
    if NumToken( idf , ';' ) > 0 then
    begin // ein oder mehr 6
      for i := 1 to NumToken( idf, ';' ) do
        StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
      Meldung( 'Ergebnis:' );
      VersetzungsMeldung( MsgNeg, msg );
      FAbgangsart := '0A';
      FBildungsgang := 'B';
      AbschlussSpeichern( 2 );
      FBildungsgang := 'A';
      AbschlussSpeichern( 2 );
      exit;
    end;

  // Anzahl Noten = 5 > 2?
    idf := NoteVorhanden( fC0, 'Note', '=', 5, -1 );
    if NumToken( idf , ';' ) >=2  then      // ggfs. Nachprüfung?
    begin // zwei oder mehr 5
      for i := 1 to NumToken( idf, ';' ) do
        StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
      Meldung( 'Ergebnis:' );
      VersetzungsMeldung( MsgNeg, msg );
      FAbgangsart := '0A';  // kein Abschluss
      FBildungsgang := 'B';
      AbschlussSpeichern( 2 );
      FBildungsgang := 'A';
      AbschlussSpeichern( 2 );
      exit;
    end;

// Wenn wir hier sind, hat der Schüler den BA bestanden

    DSN := BK_NotenDurchschnitt( fC0, 0, 'Note' );

    DurchschnittsNoteSpeichern( DSN, 'Durchschnittsnote', 'DSN_Text' );

    Meldung( 'Ergebnis:' );

  // Berufsbezogenen Abschluss speichern
{    FBildungsGang := 'B';
    fTeilPruefOrdKrz := 'APO-BK-03/A-FK/BSA';
    FAbgangsart := '3D';    // Berufschulabscluss erreicht
    AbschlussSpeichern( 1 );}
//    AbschlussErgebnisSpeichern( bwIgnorieren, bwBestanden, '', 'APO-BK-03/A-FK/BSA', '3D' );

    abschl := POJahrErsetzen( 'APO-BK-15/A-FK/BSA' );

// A: Abgangszeugnis ohne AbschlussAbgangszeugnis ohne Abschluss
// B: Hauptschulabschluss
// C: Hauptschulabschluss mit Berechtigung zum Besuch der Klasse 10, Typ B

//    if ( FLSEntlassart = ''  ) or ( FLSEntlassart = 'A' ) or ( FLSEntlassart = 'B' ) or
//       ( FLSEntlassart = 'C' ) or ( FLSEntlassart = 'M' ) or ( FLSEntlassart = 'N' ) then
// Eingangsquali niedriger als derzeit erzielte
//      AbschlussErgebnisSpeichern( bwIgnorieren, bwBestanden, '', abschl, '' ) // hier noch
kein Stat.-Kürzel rein
//    else
//      AbschlussErgebnisSpeichern( bwIgnorieren, bwBestanden, '', abschl, '3A' );

    Meldung( Format( '%s (Durchschnittsnote %s)', [ 'Berufsschulabschluss', FloatToStr( DSN
    ) ] ) );
```

```
  // Jetzt den allg.-bildeneden Abschluss in Abh. der Eingangsvoraussetzung speichern
    FBildungsGang := 'A';
    FBerufsabschluss := true;;
    cmd := Format( 'update Schueler set Berufsabschluss=%s where ID=%d', [ QuotedStr( '+' ),
    fS_ID ] );
    TransactionHandler.DoExecute( cmd );

//    if ( DSN <= 3 ) and ( FeldWertIDs( fC0, 'Fachgruppe', 'FS', 0 ) <> '' ) and
FBerufsabschluss then // Fremdsprache vorhanden

// NEU: Fallunterscheidungen
//    if ( FLSEntlassart = ''  ) or ( FLSEntlassart = 'A' ) or ( FLSEntlassart = 'B' ) or
//       ( FLSEntlassart = 'C' ) or ( FLSEntlassart = 'M' ) or ( FLSEntlassart = 'N' ) then
//    begin // Eingangsquali niedriger als derzeit erzielte
      if ReferenzNiveauVorhanden( 'B1' ) then
      begin // Englisch ausreichend vorhanden
        if DSN <= 2.5 then
        begin
          abschl := POJahrErsetzen( 'APO-BK-15/A-FK/FGO' );
          AbschlussErgebnisSpeichern( bwBestanden, bwIgnorieren, abschl, '', '3G',
          'Zusatzqualifikation FOR mit Q.V.' );
//          Meldung( 'Zusatzqualifikation FOR mit Q.V.' );
        end else if DSN <= 3.5 then
        begin
          abschl := POJahrErsetzen( 'APO-BK-15/A-FK/FOR' );
          AbschlussErgebnisSpeichern( bwBestanden, bwIgnorieren, abschl, '',
          '3F','Zusatzqualifikation FOR ohne Q.V.' );
//          Meldung( 'Zusatzqualifikation FOR ohne Q.V.' );
        end else
        begin
          abschl := POJahrErsetzen( 'APO-BK-15/A-FK/HSA' );
          AbschlussErgebnisSpeichern( bwBestanden, bwIgnorieren, abschl, '', '3D','' );
          Meldung( 'Keine Zusatzqualifikation' );
        end;
      end else
      begin  // Englisch nicht ausreichend vorhanden
        if DSN <= 2.5 then
        begin
          abschl := POJahrErsetzen( 'APO-BK-15/A-FK/HSA' );
          AbschlussErgebnisSpeichern( bwBestanden, bwIgnorieren, abschl, '', '3D','' );
          Meldung( 'Keine Zusatzqualifikation, bei Nachweis der notwendigen
          Englischkenntnisse ist FOR-Q möglich!' );
        end else if DSN <= 3.5 then
        begin
          abschl := POJahrErsetzen( 'APO-BK-15/A-FK/HSA' );
          AbschlussErgebnisSpeichern( bwBestanden, bwIgnorieren, abschl, '', '3D', '' );
          Meldung( 'Keine Zusatzqualifikation, bei Nachweis der notwendigen
          Englischkenntnisse ist FOR möglich!');
        end else
        begin
          abschl := POJahrErsetzen( 'APO-BK-15/A-FK/HSA' );
          AbschlussErgebnisSpeichern( bwBestanden, bwIgnorieren, abschl, '', '3D', '' );
          Meldung( 'Keine Zusatzqualifikation' );
        end;
      end;
//    end else
//    begin
// Bei höherer Eingansquali gar nichts tun
//    end;
    Result := 'A';
  finally
// Neu 30.1.2011: Diff-Bereich nur temporär entfernen
    Uebertragen( fC1, fC0, diff_ber );      // Diff wieder zurück
  end;
end;


function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_A_VBG_BSP_Abschluss;
```

```
// Bildungsgang A05
const
    MsgNeg = 'Anforderungen des Bildungsganges nicht erfüllt:';
    MsgPos = 'Anforderungen des Bildungsganges erfüllt! Eingangsvoraussetzung für BGJ
    erfüllt!';
var
    msg, idf: string;
    i : integer;
  erfolg_ab, erfolg_ba: TBestehungswert;
  abschl_ab, abschl_ba, stat_krz: string;
begin
    Result := 'A';
    msg := '';

    if fC0.RecordCount = 0 then
    begin
        Meldung( 'Zu wenig Fächer' );
        exit;
    end;

// Fachpraxis prüfen
{   if FeldWertAnzahl( fC0, 'FachIntern', 'FP' ) = 0 then
    begin
        Meldung( 'Fachpraxis fehlt' );
        exit;
    end; else
        Uebertragen( fC0, fC1, FeldWertIDs( fC0, 'Fach', 'FP', 0 ) );}

// Anzahl Noten = 6 prüfen
    Meldung( ' ' );
    Meldung( 'Anforderungen erfüllt?' );
    Meldung( '-----------------------------------------------------------------------' );
    idf := NoteVorhanden( fC0, 'Note', '=', 6, -1 );
    if NumToken( idf , ';' ) > 0 then
    begin // ein oder mehr 6
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        VersetzungsMeldung( MsgNeg, msg );
        Result := 'N';
    end else
    begin
// Anzahl Noten = 5 prüfen
        idf := NoteVorhanden( fC0, 'Note', '=', 5, -1 );
        if NumToken( idf , ';' ) > 1 then
        begin // zwei oder mehr 5
            for i := 1 to NumToken( idf, ';' ) do
                StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
            VersetzungsMeldung( MsgNeg, msg );
            Result := 'N';
        end else
        begin // Fachpraxis > 4 ?
            idf := Pruefe_FP( fC0, 'Note', 4 );
            if idf <> '' then
            begin // Fachpraxis 5 oder schlechter
                msg := 'Fachpraxis nicht ausreichend';
                VersetzungsMeldung( MsgNeg, msg );
                Result := 'N';
            end;
        end;
    end;

    if Result = 'N' then
    begin   // nicht bestanden
    AbschlussErgebnisSpeichern( bwNichtBestanden, bwNichtBestanden, '', '', '0A' );
        exit; // und tschüss
    end else
    begin
    erfolg_ba := bwBestanden;
```

```
    abschl_ba := POJahrErsetzen( 'APO-BK-15/A-VBG/BSP' );
{       FBildungsgang := 'B';
        FAbgangsart := '-'; // noch keine Abgangsart setzen
//      fTeilPruefOrdKrz := 'APO-BK-03/A-VBG/BSP';
        fTeilPruefOrdKrz :=
        AbschlussSpeichern( 1 );}
    end;

    Meldung( MsgPos );

    FBildungsgang := 'A';
    if Pruefe_APO_BK_15_A_VBG_HSA_Abschluss = 'A' then
    begin       // Hauptschulabschluss erreicht
    stat_krz := '1B';
    erfolg_ab := bwBestanden;
    abschl_ab := POJahrErsetzen( 'APO-BK-15/A-VBG/HSA' );
{       FAbgangsart := '1B';
//      fTeilPruefOrdKrz := 'APO-BK-03/A-VBG/HSA';
        fTeilPruefOrdKrz :=
        AbschlussSpeichern( 1 );}
    end else
    begin
    stat_krz := '1A';
    erfolg_ab := bwNichtBestanden;
    abschl_ab := '';
//      FAbgangsart := '1A';
//      AbschlussSpeichern( 2 );
    end;

  AbschlussErgebnisSpeichern( erfolg_ab, erfolg_ba, abschl_ab, abschl_ba, stat_krz );

end;

function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_A_VBG_HSA_Abschluss;
// in A05 und A07
// NEU:
const
    MsgNeg = 'Hauptschulabschluss nicht erreicht:';
    MsgPos = 'Hauptschulabschluss erreicht!';
var
    idHF, msg, idf, idf1: string;
    i : integer;
    DSN: double;
    nw_not, en_not: double;
begin
    Result := 'N';
    msg := '';

// Deutsch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'D' );
  if idHF = '' then
    begin
        Meldung( 'Deutsch fehlt' );
        exit;
    end;

// Englisch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'E' );
  if idHF = '' then
    begin
        Meldung( 'Englisch fehlt' );
        exit;
    end;

// Politik prüfen
  idHF := HauptfachID( fC0, 'Fach', 'PK' );
  if idHF = '' then
    begin
```

```
        Meldung( 'Politik fehlt' );
        exit;
    end;

// Mathematik prüfen
  idHF := HauptfachID( fC0, 'Fach', 'M' );
  if idHF = '' then
    begin
        Meldung( 'Mathematik fehlt' );
        exit;
    end;

    Meldung( ' ' );
    Meldung( 'Berechnung der allgemeinen Qualifikation (Hauptschulabschluss)' );
    Meldung(
    '-------------------------------------------------------------------------------------------
    --------------------' );

// Naturw. prüfen
    if ( FeldWertAnzahl( fC0, 'FachgruppeIntern2', 'NW' ) = 0 ) then
    begin
        Meldung( 'Naturwissenschaftliches Fach fehlt' );
        exit;
    end;

// Note Englisch mind. ausreichend

    idf := BesteNoteID( fC0, 'Fach', 'E', -1 );  // Note schlechter als 4
    if fBesteNote > 4 then
        Uebertragen( fC0, fC1, idf );   // in fC1 übertragen

    DSN := BK_NotenDurchschnitt( fC0, 3, 'Note' );

    DurchschnittsNoteSpeichern( DSN, 'Durchschnittsnote', 'DSN_Text' );
  DurchschnittsNoteSpeichern( DSN, 'DurchschnittsnoteFHR', 'DSN_FHR_Text' );

    if DSN > 4 then
    begin
        msg := 'Durchschnittsnote schlechter als 4';
        Meldung( 'Ergebnis:' );
        VersetzungsMeldung( MsgNeg, msg );
        exit;
    end;

// Deutsch prüfen
    idf := BesteNoteID( fC0, 'Fach', 'D', -1 );
    if fBesteNote > 4 then
    begin
        msg := 'Deutsch schlechter als 4';
        Meldung( 'Ergebnis:' );
        VersetzungsMeldung( MsgNeg, msg );
        exit;
    end;

// Politik prüfen
    idf := BesteNoteID( fC0, 'Fach', 'PK', -1 );
    if fBesteNote > 4 then
    begin
        msg := 'Politik schlechter als 4';
        Meldung( 'Ergebnis:' );
        VersetzungsMeldung( MsgNeg, msg );
        exit;
    end;

// Mathe prüfen
    idf := BesteNoteID( fC0, 'Fach', 'M', -1 );
    if fBesteNote > 4 then
    begin
```

```
        msg := 'Mathematik schlechter als 4';
        Meldung( 'Ergebnis:' );
        VersetzungsMeldung( MsgNeg, msg );
        exit;
    end;

// Naturw. prüfen
    idf := BesteNoteID( fC0, 'FachgruppeIntern2', 'NW', -1 );
    nw_not := fBesteNote;

// Englisch prüfen
    if fC1.RecordCount = 0 then
        idf := BesteNoteID( fC0, 'Fach', 'E', -1 )
    else
        idf := BesteNoteID( fC1, 'Fach', 'E', -1 );
    en_not := fBesteNote;

    if ( en_not > 4 ) and ( nw_not > 4 ) then
    begin
        msg := 'Englisch und naturwissenschaftliches Fach schlechter als 4';
        Meldung( 'Ergebnis:' );
        VersetzungsMeldung( MsgNeg, msg );
        exit;
    end;

    Result := 'A';
    Meldung( 'Ergebnis:' );
    Meldung( Format( '%s (Durchschnittsnote %s)', [MsgPos, FloatToStr( dsn ) ] ) );

end;

// Bildungsgang A06, weitestgehend identsich mit B06

function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_A_BG_Abschluss: string;
const
    MsgNeg = 'Anforderungen des Bildungsganges nicht erfüllt:';
    MsgPos = 'Berufliche Grundbildung erreicht!';
var
    idHF, msg, idf: string;
    i : integer;
    dn : double;
  erfolg_ab, erfolg_ba: TBestehungswert;
  abschl_ab, abschl_ba, stat_krz: string;
begin
    Result := 'A';
    msg := '';

// Differenzierungsbereich raus
    AusContainerLoeschen( fC0, FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 ) );

    if fC0.RecordCount = 0 then
    begin
        Meldung( 'Zu wenig Fächer' );
        exit;
    end;

// Deutsch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'D' );
  if idHF = '' then
    begin
        Meldung( 'Deutsch fehlt' );
        exit;
    end;

// Englisch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'E' );
  if idHF = '' then
    begin
```

```
        Meldung( 'Englisch fehlt' );
        exit;
    end;

// Mathematik prüfen
  idHF := HauptfachID( fC0, 'Fach', 'M' );
  if idHF = '' then
    begin
        Meldung( 'Mathematik fehlt' );
        exit;
    end;

// Anzahl Noten = 6 prüfen
    Meldung( ' ' );
    Meldung( 'Anforderungen erfüllt?' );
    Meldung( '-------------------------------------------------------------------' );
    idf := NoteVorhanden( fC0, 'NoteAbschl', '=', 6, -1 );
    if NumToken( idf , ';' ) > 0 then
    begin // ein oder mehr 6
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        VersetzungsMeldung( MsgNeg, msg );
        Result := 'N';
    end else
    begin
// Anzahl Noten = 5 prüfen
        idf := NoteVorhanden( fC0, 'NoteAbschl', '=', 5, -1 );
        if NumToken( idf , ';' ) > 1 then
        begin // zwei oder mehr 5
            for i := 1 to NumToken( idf, ';' ) do
                StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
            VersetzungsMeldung( MsgNeg, msg );
            Result := 'N';
        end;
    end;

    if Result = 'N' then
    begin   // nicht bestanden
    AbschlussErgebnisSpeichern( bwNichtBestanden, bwNichtBestanden, '', '', '0A' );
        exit; // und tschüss
    end else
    begin
    erfolg_ba := bwBestanden;
    case FBKGLiederung of
    glA06:
      begin
        abschl_ba := 'APO-BK-11/A-BG/BG';
        stat_krz := '2A'; // ??? wird doch später eh neu bestimmt
      end;
    glB06:
      begin
        erfolg_ab := bwBestanden;
        abschl_ba := 'APO-BK-15/B/BK';
        abschl_ab := 'APO-BK-15/B/HSA';
        stat_krz := '4D';
        Meldung( 'Allgemeine Qualifikation: HSA Klasse 10');
        AbschlussErgebnisSpeichern( erfolg_ab, erfolg_ba, abschl_ab, abschl_ba, stat_krz );
      end;
    end;
  end;

    Meldung( MsgPos );

  if FBKGLiederung = glB06 then
    exit;


// Jetzt die Berechnung der allg. Qualifikation  bei A06
```

```
    FBildungsgang := 'A';

// Deutsch , Mathe und Englisch in fC1
    Uebertragen( fC0, fC1, HauptfachID( fC0, 'Fach', 'D' ) );
  Uebertragen( fC0, fC1, HauptfachID( fC0, 'Fach', 'M' ) );
    Uebertragen( fC0, fC1, HauptfachID( fC0, 'Fach', 'E' ) );

    dn := 0;
    fC1.First;
    while not fC1.EOF do
    begin
        dn := dn + fC1.FieldByname( 'NoteAbschl' ).AsFloat;
        fC1.Next;
    end;

    dn := ( int( dn/3*10 ) )*0.1;   // auf eine Nachkommastalle, nicht runden

    DurchschnittsNoteSpeichern( dn, 'Durchschnittsnote', 'DSN_Text' );
  DurchschnittsNoteSpeichern( dn, 'DurchschnittsnoteFHR', 'DSN_FHR_Text' );

  erfolg_ab := bwBestanden;

  if dn <= 3 then
  begin     // FOR-Qualifikation
    abschl_ab := 'APO-BK-11/A-BG/FOR';
    stat_krz := '2F';
    Meldung( 'Allgemeine Qualifikation: FOR');
  end else
  begin
    abschl_ab := 'APO-BK-11/A-BG/SI';
    stat_krz := '2D';
    Meldung( 'Allgemeine Qualifikation: HSA Klasse 10');
  end;

  AbschlussErgebnisSpeichern( erfolg_ab, erfolg_ba, abschl_ab, abschl_ba, stat_krz );

end;



/// Bildungsgang A07

function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_A_JOA: string;
// Bildungsgang Jugendliche ohne Ausbildungsverhältnis A07
const
    MsgNeg = 'Anforderungen des Bildungsganges nicht erfüllt:';
    MsgPos = 'Anforderungen des Bildungsganges erfüllt!';
var
    msg, idf: string;
    i : integer;
    DSN : Double;
    ok : boolean;
  erfolg_ab, erfolg_ba: TBestehungswert;
  abschl_ab, abschl_ba, stat_krz: string;
begin
    Result := 'N';
    msg := '';

// Anforderungen des Bildungsganges erfüllt?
    ok := true;

//  if fAnzLeer > 0 then
//      Meldung( 'Hinweis: Nicht alle Fächer benotet' );

// Anzahl Noten = 6 > 0?
    idf := NoteVorhanden( fC0, 'Note', '=', 6, -1 );
    if NumToken( idf , ';' ) > 0 then
    begin // ein oder mehr 6
```

```
        ok := false;
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
    end else
    begin
// Anzahl Noten = 5 >= 2?
        idf := NoteVorhanden( fC0, 'Note', '=', 5, -1 );
        if NumToken( idf , ';' ) >=2  then
        begin // zwei oder mehr 5
            ok := false;
            for i := 1 to NumToken( idf, ';' ) do
                StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        end;
    end;

    FBildungsGang := 'B';       // Berufsbildend
    if not ok then
    begin   // an schlechten Noten gescheitert
        VersetzungsMeldung( MsgNeg, msg );
    erfolg_ba := bwNichtBestanden;
    abschl_ba := '';
//      AbschlussSpeichern( 2 );
    end else
    begin
        Meldung( MsgPos );
    erfolg_ba := bwBestanden;
    abschl_ba := POJahrErsetzen( 'APO-BK-15/A-JOA/BK' );
//      fTeilPruefOrdKrz :=  'APO-BK-11/A-JOA/BK';//'APO-BK-03/A-JOA/BK';           // Ist
das richtig ? BSP: Erfüllung der Berufsschulpflicht nach APO-BK-03, Anl. A, VV zu § 22
//      AbschlussSpeichern( 1 );
    end;

    FBildungsGang := 'A';       // Jetzt allgemein-bildend
    if FJahrgang = '01' then
    begin       // Versetzung obligatorisch
    erfolg_ab := bwIgnorieren;
    abschl_ab := POJahrErsetzen( 'APO-BK-15/A-JOA/V11' );
    stat_krz := '';

{       FAbgangsart := '-';
        fTeilPruefOrdKrz := 'APO-BK-11/A-JOA/V11';//'APO-BK-03/A-JOA/V11';
    if ok then
        AbschlussSpeichern( 1 )
    else
      AbschlussSpeichern( 2 );}
    end else if FJahrgang = '02' then
    begin
        if ok and ( Pruefe_APO_BK_15_A_VBG_HSA_Abschluss = 'A' ) then
        begin       // Hauptschulabschluss erreicht
      erfolg_ab := bwBestanden;
      abschl_ab := POJahrErsetzen( 'APO-BK-15/A-JOA/HSA' );
      stat_krz := '1B';
//          FAbgangsart := '1B';        // Abschlusszeugnis und Hauptschulabschluss
//          fTeilPruefOrdKrz := 'APO-BK-11/A-JOA/HSA';//'APO-BK-03/A-JOA/HSA';
//          AbschlussSpeichern( 1 );
        end else
        begin   // kein HSA
      erfolg_ab := bwNichtBestanden;
      abschl_ab := '';
      stat_krz := '1A';
//          FAbgangsart := '1A';        // Abschlusszeugnis oder 0A = kein Abschluss
//          AbschlussSpeichern( 2 );
        end;
    end;

  AbschlussErgebnisSpeichern( erfolg_ab, erfolg_ba, abschl_ab, abschl_ba, stat_krz );

end;
```

```
function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_A12_A13: string;
const
    MsgNeg = 'Hauptschulabschluss nicht erreicht:';
    MsgPos = 'Hauptschulabschluss erreicht!';
var
  msg: string;

  procedure AbschlussEintragen( const best: boolean );
  var
    erfolg_ab, erfolg_ba: TBestehungswert;
    abschl_ab, abschl_ba, stat_krz: string;
  begin

       Meldung( 'Ergebnis:' );

    if best then
    begin
       VersetzungsMeldung( MsgPos, msg );
      erfolg_ab := bwBestanden;
      abschl_ab := 'APO-BK-15/A-AV/HS';

      erfolg_ba := bwBestanden;
      abschl_ba := 'APO-BK-15/A-AV/BK';
      stat_krz := '1B';
    end else
    begin
       VersetzungsMeldung( MsgNeg, msg );
      erfolg_ab := bwNichtBestanden;
      abschl_ab := '';

      erfolg_ba := bwNichtBestanden;
      abschl_ba := '';
      stat_krz := '0A';
    end;
    AbschlussErgebnisSpeichern( erfolg_ab, erfolg_ba, abschl_ab, abschl_ba, stat_krz );

  end;

var
    idM, idNW, idf: string;
    i : integer;
    DSN: double;
    nw_not, m_not: double;
  nw_wst, m_wst: integer;
  ok: boolean;
begin // NEU Juni 2016 für A12 und A13
// In Allen Fächern 4 oder besser
// Wenn M=5 dann Hinweis
    ResetBKAbschluss( fS_ID );
  Result := 'N';
    msg := '';

// Mathematik prüfen
  idM := HauptfachID( fC0, 'Fach', 'M' );
  if idM = '' then
    begin
       Meldung( 'Mathematik fehlt' );
       exit;
    end;

//  Meldung( ' ' );
//  Meldung( 'Berechnung der allgemeinen Qualifikation (Hauptschulabschluss)' );
//  Meldung(
'-------------------------------------------------------------------------------------------
---------------' );
```

```
// Prüfen, ob alle Noten 4 oder besser
    idf := NoteVorhanden( fC0, 'Note', '>', 4, -1 );
    ok := AnzahlElemente( idf) = 0;

  if ok then
  begin // alles in Ordnung
        msg := 'Keine Note schlechter als 4';
    AbschlussEintragen( true );
    Result := 'A';
    exit;
  end;

// Wenn wir hier sind, sind Noten schlechter als 4

// Englisch ignorieren
  idf := HauptfachID( fC0, 'Fach', 'E' );
  if not IstLeer( idf ) then
    Uebertragen( fC0, fC1, idf );   // in fC1 übertragen

// Prüfe NW = 5, ignorieren
  idf := HauptfachID( fC0, 'FachgruppeIntern2', 'NW' );   // alle Naturwissenschaften
  idf := NoteVorhandenAusListe( FC0, idf, 'Note', '>', 4 );
  if not IstLeer( idf ) then
    Uebertragen( fC0, fC1, idf );   // in fC1 übertragen

// Eine weitere 5 ignoriern, aber nur, wenn nicht Mathe
  idf := NoteVorhanden( fC0, 'Note', '>', 4, -1 );
  if not IstLeer( idf ) then
  begin
    AusMengeLoeschen( idf, idM );
    if not IstLeer( idf ) then
    begin
      idf := EinzelElement( idf, 1 );
      Uebertragen( fC0, fC1, idf ); // in fC1 übertragen
    end;
  end;

// 5 in Mathe?
  idf := NoteVorhandenAusListe( FC0, idM, 'Note', '>', 4 );
  if not IstLeer( idf ) then
  begin // 5 in Mathe, gibt es in NW Note 4 oder beseer
    idNW := HauptfachID( fC0, 'FachgruppeIntern2', 'NW' );   // alle Naturwissenschaften
    idNW := NoteVorhandenAusListe( FC0, idNW, 'Note', '<', 5 );
    if not IstLeer( idNW ) then
    begin // potentielles Ausgleichsfach, aber: Ist Wochsnetundenzahl gleich mit der von M
      idM := EinzelElement( idM, 1 );
      FC0.Locate( 'ID', StrToInt( idM ), [] );
      m_wst := FC0.FieldByName( 'Wochenstd' ).AsInteger;
      for i := 1 to AnzahlElemente( idNW ) do
      begin
        idf := EinzelElement( idNW, i );
        FC0.Locate( 'ID', StrToInt( idf ), [] );
        nw_wst := FC0.FieldByName( 'Wochenstd' ).AsInteger;
        if nw_wst = m_wst then
        begin
          Meldung( Format( 'Nicht ausreichende Leistung in Mathematik, Ausgleichsfach %s
          (mit gleichem Stundenumfang)', [ FC0.FieldByName( 'FachName' ).AsString ] ) );
          Result := 'A';
          break;
        end;
      end;
    end;
    if Result = 'A' then
      Uebertragen( fC0, fC1, idM ); // M in fC1 übertragen
  end;

// Jetzt noch prüfen, ob noch 5en da
  idf := NoteVorhanden( fC0, 'Note', '>', 4, -1 );
```

```
  ok := AnzahlElemente( idf ) = 0;
  if not ok then
  begin
    Result := 'N';
    for i := 1 to AnzahlElemente( idf ) do
      StrAdd( SchreibeFachNoteFeld( fC0, StrToInt( EinzelElement( idf, i ) ), 'Note' ), msg );
  end else
    Result := 'A';

  AbschlussEintragen( Result = 'A' );


end;

//////// Bildungsgang B01

// Versetzung
function TBKPruefungsAlgorithmus.Pruefe_BK_B_Versetzung: string;
var
    i : integer;
    idf, msg: string;
    sum: double;
begin
    Result := 'N';
    msg := '';

// Anzahl 6
    idf := NoteVorhanden( fC0, 'Note', '=', 6, -1 );
    if idf <> '' then
    begin       // 6 vorhanden
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        VersetzungsMeldung( 'Keine Versetzung:', msg );
        exit;
    end;

    idf := NoteVorhanden( fC0, 'Note', '=', 5, -1 );
    if NumToken( idf , ';' ) > 1 then  // zwei oder mehr 5
    begin
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        VersetzungsMeldung( 'Keine Versetzung:', msg );
        exit;
    end;

    idf := Pruefe_FP( fC0, 'Note', 4 );     // Fachpraxis prüfen
    if idf <> '' then
    begin
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        VersetzungsMeldung( 'Keine Versetzung:', msg );
        exit;
    end;

    Result := 'V';
    Meldung( 'Versetzt!' );

// Jetzt Mittelwert von D,M,E berechnen
    if fNachprFaecherErmitteln then
        exit;

    sum := 0;
    Uebertragen( fC0, fC1, HauptfachID( fC0, 'Fach', 'D' ) );
    Uebertragen( fC0, fC1, HauptfachID( fC0, 'Fach', 'M' ) );
    Uebertragen( fC0, fC1, HauptfachID( fC0, 'Fach', 'E' ) );
    if fC1.RecordCount <> 3 then
    begin
        Meldung( 'Deutsch, Mathematik oder Englisch fehlt' );
```

```
        exit;
    end;
    fC1.First;
    while not fC1.Eof do
    begin
        sum := sum + fC1.FieldByName( 'Note' ).AsFloat;
        fC1.Next;
    end;
    sum := sum / 3;

    if sum <= 3 then
    begin
// Feststellen, ob M oder E Grund- oder Erweiterungskurs ist (nur für Fachklasse
Kinderpfleger)
        if BK_B_KinderpflegerEKurs( fJahr, true ) then
            FAbgangsart := '5F'     // SI FOR
        else
            FAbgangsart := '5A';        // SI 10A
    end else// Quali 10A erreicht
        FAbgangsart := '5A';

    AbschlussSpeichern( 1 );

end;

function TBKPruefungsAlgorithmus.BK_B_MuendlichePruefungZulaessig( ds: TDataSet ): boolean;
var
    idf, msg: string;
    i : integer;
begin
    idf := NoteVorhanden( ds, 'Note', '>', 4, -1 );
    Result := NumToken( idf, ';' ) < 4;
    if not Result then
    begin
        msg := '';
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( ds, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        VersetzungsMeldung( 'Keine Zulassung zur mündlichen Prüfung:' , msg );
    end;
end;

function TBKPruefungsAlgorithmus.BK_QVermerkBerechnen: boolean;
var
    n_E, n_M, n_D, n_ds: double;
    idf, M : string;
    anz_4: integer;
begin
    anz_4 := 0;

    fC0.Locate( 'Fach', 'D', [] );
    n_D := fC0.FieldByname( 'Note' ).AsFloat;
    ZuMengeHinzu( M, fC0.FieldByName( 'ID' ).AsString );
    if n_D >= 4 then
        inc( anz_4 );

    fC0.Locate( 'Fach', 'M', [] );
    n_M := fC0.FieldByname( 'Note' ).AsFloat;
    ZuMengeHinzu( M, fC0.FieldByName( 'ID' ).AsString );
    if n_M >= 4 then
        inc( anz_4 );

    fC0.Locate( 'Fach', 'E', [] );
    n_E := fC0.FieldByname( 'Note' ).AsFloat;
    ZuMengeHinzu( M, fC0.FieldByName( 'ID' ).AsString );
    if n_E >= 4 then
        inc( anz_4 );

    if ( n_D <= 2 ) and ( n_M <= 2 ) and ( n_E <= 2 ) then
```

```
        Result := true
    else
    begin
        n_ds := ( n_D + n_M + n_E ) / 3;
        idf := NoteVorhanden( fC0, 'Note', '<=', 3, -1 );
        M := DifferenzMenge( idf, M );
        Result := ( n_ds <= 3 ) and     // Durchschnitt besser als 3
                  ( NumToken( M, ';' ) >= 3 ) and //  3 Fächer oder mehr mit <= 3
                  ( n_D <= 4 ) and ( n_M <= 4 ) and ( n_E <= 4 ) and
                  ( anz_4 <= 1 );
    end;

end;

procedure TBKPruefungsAlgorithmus.BK_B_AbschlussnotenBerechnen( ds: TDataset; const
gew_feld: string );
// Derzeit nur für B01 verwendet
var
    anot, vnot: integer;
  bue_not, ba_not: integer;
  gw_vn: integer;
begin
// Absckussberechnung für die Gliederungen B01, C01, C02, C03, C04, C07, C08
// wenn FSchulgliederung = C01:
// Wenn sich nach Durchführung eine mdl. Prüfung die Endnote verschlechtert wird diese
Änderung nur für die jeweilige Teilprüfung
// übernommen, die Abschlussnote für die zweite Teilprüfung bleibt unverändert. In diesem
Fall müssen 2 Abschkussnoten gespeichert werden

  if FBKGliederung in [ glC01, glC02, glC03, glC04, glC07, glC08 ] then
    gw_vn := 1
  else
    gw_vn := 2; // für BO1 ???
  ds.First;
    while not ds.EOF do
    begin
//    ShowMessage( ds.FieldByName( 'FachKrz' ).AsString );
//    if ds.FieldByname( gew_feld ).AsInteger > 0 then // Gewichtung hier notwendig?
    begin
      vnot := trunc( ds.FieldByName( 'Note' ).AsFloat );  // die Vornote
      if not ds.FieldByName( 'NoteMdl' ).IsNull then
      begin // es existiert eine mündliche Note
        if not ds.FieldByName( 'NoteSchr' ).IsNull then
        begin // es exitsiert eine schriftliche Note
          anot := round( ( gw_vn*ds.FieldByName( 'Note' ).AsFloat + ds.FieldByName(
          'NoteSchr' ).AsFloat + ds.FieldByname( 'NoteMdl' ).AsFloat ) / ( gw_vn + 2 ) );
        end else    // es ex. keine schriftliche Note
        begin
          anot := round( ( gw_vn*ds.FieldByName( 'Note' ).AsFloat + ds.FieldByname(
          'NoteMdl' ).AsFloat ) / ( gw_vn + 1 ) );
        end;
      end else if not ds.FieldByName( 'NoteSchr' ).IsNull then
      begin // es existiert nur eine schriftliche Note
        anot := round( ( gw_vn*ds.FieldByName( 'Note' ).AsFloat + ds.FieldByname( 'NoteSchr'
        ).AsFloat ) / ( gw_vn + 1 ) );
      end else
      begin // Nur Vornote
        anot := round( ds.FieldByName( 'Note' ).AsFloat );
      end;

      if ds.FieldByName( 'Note' ).IsNull then // Sonderfall für Abschlussartbeit
        anot := ds.FieldByname( 'NoteSchr' ).AsInteger;

      if not( FBKGliederung in [ glC01, glC02 ] ) then
      begin

        if ( ds.FieldByname( 'NoteAbschl' ).IsNull ) or ( ds.FieldByname( 'NoteAbschl'
        ).AsInteger = 999 ) then
```

```
        begin
          ds.Edit;
          ds.FieldByname( 'NoteAbschl' ).AsInteger := anot;
          ds.Post;

          TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKFaecher SET
          NoteAbschluss=%s, NoteAbschlussBA=null WHERE Schueler_ID=%d AND Fach_ID=%d',
                                    [ QuotedStr( IntToStr( anot ) ), fS_ID, ds.FieldByName(
                                    'Fach_ID' ).AsInteger ] ) );
        end;
      end else
      begin
  // Bei C01 muss überprüft werden, ob sich durch die mündliche Prüfung das Ergebnis
  verschlechtert hat
        if FBKGliederung = glC01 then
        begin
          if ( ds.FieldByName( 'IstSchriftlichBA' ).AsString = '+' ) and ( ds.FieldByName(
          'IstSchriftlich' ).AsString = '+' ) then
          begin // Fach für BÜ und BA schriftlich (kommt das überhaupt vor?)
            bue_not := anot;
            ba_not := anot;
          end else if ( ds.FieldByName( 'IstSchriftlichBA' ).AsString = '+' ) then
          begin
            ba_not := anot;
            if vnot < anot then
              bue_not := vnot // Note für FHR würde sich durch mdl. BA-Prüfung
              verschlechtern, also Note ohne die mdl. Prüfung nehmen
            else
              bue_not := anot;
          end else if ( ds.FieldByName( 'IstSchriftlich' ).AsString = '+' ) then
          begin
            bue_not := anot;
            if vnot < anot then
              ba_not := vnot  // Note für BA würde sich durch mdl. FHR-Prüfung
              verschlechtern, also Note ohne die mdl. Prüfung nehmen
            else
              ba_not := anot;
          end else
          begin
            bue_not := anot;
            ba_not := anot;
          end;
        end else
        begin // Bei C02 beide Noten gleich
          bue_not := anot;
          ba_not := anot;
        end;
        ds.Edit;
        ds.FieldByname( 'NoteAbschl' ).AsInteger := bue_not;
        ds.FieldByname( 'NoteAbschlBA' ).AsInteger := ba_not;
        ds.Post;

        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKFaecher SET
        NoteAbschluss=%s, NoteAbschlussBA=%s WHERE Schueler_ID=%d AND Fach_ID=%d',
                                  [ QuotedStr( IntToStr( bue_not ) ), QuotedStr( IntToStr(
                                  ba_not ) ), fS_ID, ds.FieldByName( 'Fach_ID' ).AsInteger ]
                                  ) );
      end;
    end;
        ds.Next;
    end;
end;


function TBKPruefungsAlgorithmus.IstSpezielleFachklasse( const fkl_liste: string ): boolean;
var
    qry: TBetterADODataset;
begin
```

```
    Result := false;
    if FFachklasse_ID = 0 then
        exit;

    qry := CreateForwardDataset( fConSchild, SchildSettings.DBFormat <> 'MSACCESS' );
    try
        qry.CommandText := 'SELECT Kennung FROM EigeneSchule_Fachklassen WHERE ID=' +
        IntToStr( FFachklasse_ID );
        qry.Open;
        Result := IstInListe( qry.FieldByName( 'Kennung' ).AsString, fkl_liste );
  finally
    FreeAndNil( qry );
  end;
end;


function TBKPruefungsAlgorithmus.BK_B_KinderpflegerEKurs( const jahr: integer; const msg:
boolean ): boolean;
var
    idf: string;
    qry: TBetterADODataset;
begin
    Result := false;
    if FFachklasse_ID = 0 then
        exit;

  if IstSpezielleFachklasse( '6-134-00;60-134-00' ) then
  begin     // Kinderpfleger
    idf := '';
    if fC0.Locate( 'Fach', 'M', [] ) then
      idf := fC0.FieldByName( 'Fach_ID' ).AsString
    else if fC1.Locate( 'Fach', 'M', [] ) then
      idf := fC1.FieldByName( 'Fach_ID' ).AsString;

    if fC0.Locate( 'Fach', 'E', [] ) then
    begin
      if idf <> '' then
        idf := idf + ',';
      idf := idf + fC0.FieldByName( 'Fach_ID' ).AsString;
    end else if fC1.Locate( 'Fach', 'E', [] ) then
    begin
      if idf <> '' then
        idf := idf + ',';
      idf := idf + fC1.FieldByName( 'Fach_ID' ).AsString;
    end;

    if idf <> '' then
    begin
        qry := CreateForwardDataset( fConSchild, SchildSettings.DBFormat <> 'MSACCESS' );
      try
        qry.CommandText := Format
                    ( 'SELECT L.* FROM SchuelerLeistungsdaten L,
                    SchuelerLernabschnittsdaten A' +
                      ' WHERE A.Schueler_ID=%d AND A.Jahr=%d AND A.Abschnitt=%d AND
                      A.WechselNr=999' +
                      ' AND A.Hochrechnung=0 AND L.Hochrechnung=0 AND A.SemesterWertung=%s
                      AND L.Abschnitt_ID=A.ID AND L.Fach_ID IN (%s)' +
                      ' AND L.SchulnrEigner=%d',
                      [ fS_ID, jahr, fAbschnitt, QuotedStr( '+' ), idf,
                      SchildSettings.Schulnr ] );
        qry.Open;
        while not qry.EOF do
        begin
          Result := Result or ( qry.FieldByName( 'Kursart' ).AsString = 'FOR' );
          qry.Next;
        end;
      finally
        FreeAndNil( qry );
```

```
      end;
    end;
    if msg and not Result then
      Meldung( 'Nachholen von FOR im 2. Jahr nicht möglich' );
  end;
end;

function TBKPruefungsAlgorithmus.BK_B_ZulassungPruefen: boolean;
var
    i : integer;
    idf, msg: string;
    sErg, sNeg, sPos: string;
// bei B01, B02 (alt), B03 und B07

  function AusgleichVorhanden: boolean;
  begin
    Result :=   NoteVorhanden( fC0, 'Note', '<', 4, -1 ) <> '';
  end;

begin
    msg := '';
    Result := true;
// Alles auf 0 setzen
    ResetBKAbschluss( fS_ID );
    TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKFaecher SET MdlPruefung=%s,
    NoteMuendlich=NULL, NoteAbschluss=NULL WHERE Schueler_ID=%d',
                                          [ QuotedStr( '-' ), fS_ID ] ) );

//  if fAnzLeer > 0 then
//      Meldung( 'Hinweis: Nicht alle Fächer benotet' );
// Differenzierungsbereich raus
    AusContainerLoeschen( fC0, FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 ) );

    idf := NoteVorhanden( fC0, 'Note', '=', 6, -1 );
    if NumToken( idf , ';' ) > 0 then// Anzahl Noten = 6 > 0?
        Result := false
    else
    begin
        idf := NoteVorhanden( fC0, 'Note', '=', 5, -1 );
    case FBKGLiederung of
    glB01:
      begin
            if AnzahlElemente( idf ) >= 2 then  // NEU: zwei 5 auch?
                Result := false;
      end;
    glB02:
      begin
            if AnzahlElemente( idf ) >= 2 then  // NEU: zwei 5 auch?
                Result := false;
      end;
    glB03:
      begin
        if AnzahlElemente( idf ) >= 2 then      // bei B03 >= zwei Fünfen
          Result := false;
      end;
    glB07:
      begin
            if AnzahlElemente( idf ) >= 2 then
                Result := false;
      end;
    glB08, glB09, glB10:
      begin
        if AnzahlElemente( idf ) = 1 then // Prüfen, ob Ausgleich
          Result := AusgleichVorhanden
        else if AnzahlElemente( idf ) >= 2 then
                Result := false;
      end;
```

```
    end;

        if Result then
        begin       // FP = 5?
            idf := Pruefe_FP( fC0, 'Note', 4 );
            Result := idf = '';
        end;
    end;

  case FBKGliederung of
  glB01, glB08,glB09, glB10:
    begin
        sErg := 'Ergebnis Zulassungsprüfung';
          sNeg := 'Keine Zulassung zur Prüfung:';
          sPos := 'Zur Prüfung zugelassen';
    end;
  else
    begin
      sErg := 'Ergebnis';
      sNeg := 'Kein Abschluss, keine berufliche Grundbildung';
      sPos := 'Abschluss, berufliche Grundbildung';
    end;
  end;

    Meldung( sErg );
    Meldung( '-------------------------' );
    if not Result then
    begin
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );

        VersetzungsMeldung( sNeg , msg );
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s,
        Bestanden=%s, Zeugnis=%s WHERE Schueler_ID=%d',
                                                    [ QuotedStr( '-' ),
                                                      QuotedStr( '-' ),
                                                      QuotedStr( 'A' ),
                                                      fS_ID ] ) );
        SchuelerAbschlussSpeichern( '0A' );
    end else
    begin
        Meldung( sPos );
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s,
        Zeugnis=%s WHERE Schueler_ID=%d',
                                                  [ QuotedStr( '+' ),
                                                    QuotedStr( '?' ),
                                                    fS_ID ] ) );
    end;
end;


function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_B_BAB_Abschluss_B01_B08_B09_B10: string;
var
    msg, idf: string;


    function MuendlichePruefungen: boolean;
    var
        cnt : integer;
    begin
        Result := true;
// Schiftliche Fächer in fC1 übertragen
        Uebertragen( fC0, fC1, FeldWertIDs( fC0, 'IstSchriftlich', '+', 0 ) );
        if fC1.RecordCount <{>} 2 then
        begin
            Meldung( 'Falsche Anzahl schriftlicher Fächer' );
            Result := false;
            exit;
```

```
        end;
        if FeldWertAnzahl( fC1, 'NoteSchr', '999' ) > 0 then
        begin
            Meldung( 'Schriftliche Noten unvollständig' );
            Result := false;
            exit;
        end;

        fC1.First;
        cnt := 0;
        while not fC1.EOF do
        begin
      if not fC1.FieldByname( 'Note' ).IsNull then
      begin
        if ( ( fC1.FieldByname( 'Note' ).AsFloat = 5 ) and ( fC1.FieldByname( 'NoteSchr'
        ).AsFloat = 4 ) ) or
           ( abs( fC1.FieldByname( 'Note' ).AsFloat - fC1.FieldByname( 'NoteSchr' ).AsFloat
           ) >=2 ) then
        begin
          if msg <> '' then
            msg := msg + ';';
          msg := msg + Format( '%s: Vornote=%s, Prüfungsnote=%s',
                              [ fC1.FieldByname( 'Fachname' ).AsString,
                                fC1.FieldByname( 'Note' ).AsString,
                                fC1.FieldByname( 'NoteSchr' ).AsString ] );
          TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKFaecher SET MdlPruefung=%s
          WHERE Schueler_ID=%d AND Fach_ID=%d',
                              [ QuotedStr( '+' ), fS_ID, fC1.FieldByname( 'Fach_ID'
                              ).AsInteger ] ) );
          inc( cnt );

        end;
      end;
            fC1.Next;
        end;

        Meldung( 'Mündliche Prüfung' );
        Meldung( '--------------------------------' );
        if cnt = 0 then
        begin
            Meldung( 'Keine mündlichen Prüfungen notwendig' );
            exit;       // keine mündliche Prüfung notwendig
        end;

        VersetzungsMeldung( 'Abweichung Vornote/Prüfungsnote:' , msg );

//  Prüfen, ob mdl. Prüfung zulässig
        if not BK_B_MuendlichePruefungZulaessig( fC0 ) then
        begin
            Meldung( 'Keine mündliche Prüfung zulässig' );
            if not FWiederholung then
                TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET
                Bestanden=%s, Zeugnis=%s WHERE Schueler_ID=%d',
                                                      [ QuotedStr( '-' ),
                                                      QuotedStr( 'J' ),   fS_ID ]
                                                      ) )
            else
                TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET
                Bestanden=%s, Zeugnis=%s WHERE Schueler_ID=%d',
                                                      [ QuotedStr( '-' ),
                                                      QuotedStr( 'A' ), fS_ID ] ) );
            FAbgangsart := '0A';
            FBildungsgang := 'B';
            AbschlussSpeichern( 2 );
            FBildungsgang := 'A';
            AbschlussSpeichern( 2 );

        end else
```

```
            Meldung( 'Mündliche Prüfung zulässig' );

    end;        // end function


{  procedure AbschlussEintragen( const best: boolean; const stat_krz, abschluss: string );
  var
    erfolg_ab, erfolg_ba: TBestehungswert;
    abschl_ab, abschl_ba: string;
  begin

       Meldung( 'Ergebnis:' );

    if best then
    begin
       VersetzungsMeldung( MsgPos, msg );
      erfolg_ab := bwBestanden;
      case FBKGLiederung of
      glB01:
        begin
          if not best then
          begin
            erfolg_ab := bwNichtBestanden;
            erfolg_ba := bwNichtBestanden;
            abschl_ab := '';
            abschl_ba := '';
          end else
          begin

            POJahrErsetzen('APO-BK-15/B/BAB');//'APO-BK-03/B/BAB'
          end;

        end;
      glB08, glB09, glB10
      abschl_ab := 'APO-BK-15/A-AV/HS';

      erfolg_ba := bwBestanden;
      abschl_ba := 'APO-BK-15/A-AV/BK';
      stat_krz := '1B';
    end else
    begin
       VersetzungsMeldung( MsgNeg, msg );
      erfolg_ab := bwNichtBestanden;
      abschl_ab := '';

      erfolg_ba := bwNichtBestanden;
      abschl_ba := '';
      stat_krz := '0A';
    end;
    AbschlussErgebnisSpeichern( erfolg_ab, erfolg_ba, abschl_ab, abschl_ba, stat_krz );

  end;}


    function AbschlussBerechnen: boolean;
    var
        idf, idfa, quali: string;
        si_for: boolean;
        si_10a: boolean;
        hsa, qv: boolean;
        i : integer;
    erfolg_ab, erfolg_ba: TBestehungswert;
    abschl_ab, abschl_ba, stat_krz: string;
    dn: double;

    begin

// Schiftliche Fächer in fC1 übertragen
```

```
      Uebertragen( fC0, fC1, FeldWertIDs( fC0, 'IstSchriftlich', '+', 0 ) );
      if FeldWertAnzahl( fC1, 'IstSchriftlich', '+' ) < 2 then
      begin
          Meldung( 'Falsche Anzahl schriftlicher Fächer' );
          exit;
      end;
      if FeldWertAnzahl( fC1, 'NoteSchr', '999' ) > 0 then
      begin
          Meldung( 'Schriftliche Noten unvollständig' );
          Result := false;
          exit;
      end;
      if FeldWertAnzahl( fC1, 'NoteMdl', '999' ) > 0 then
      begin
          Meldung( 'Mündliche Noten unvollständig' );
          Result := false;
          exit;
      end;

      Result := true;
      BK_B_AbschlussnotenBerechnen( fC0, 'Gewichtung' );
      BK_B_AbschlussnotenBerechnen( fC1, 'Gewichtung' );

      Meldung ( ' ' );
      Meldung( 'Abschluss-Berechnung' );
      Meldung( '-----------------------------------------');
      idf := NoteVorhanden( fC0, 'NoteAbschl', '>=', 5, -1 );
      if NumToken( idf , ';' ) >= 2 then  // zwei oder mehr 5
      begin
          for i := 1 to NumToken( idf, ';' ) do
              StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
          VersetzungsMeldung( 'Kein Abschluss: ', msg );
          Result := false;
      end else if NumToken( idf , ';' ) = 1 then  // eine 5, prüfen ob Ausgleich vorhanden
      begin
          idfa := BesteNoteID( fC0, '', '', -1 );
          Result := ( idfa <> '' ) and ( fBesteNote < 4 );
          if not Result then
          begin
              StrAdd( SchreibeFachNote( fC0, StrToInt( idf ) ), msg );
              msg := msg + ';' + 'Kein Ausgleichsfach gefunden';
              VersetzungsMeldung( 'Kein Abschluss: ', msg );
          end;
      end;

      if Result then      // Prüfen, ob Fachpraxis <= 4
      begin
          idf := Pruefe_FP( fC0, 'NoteAbschl', 4 );
          if idf <> '' then
          begin
              Result := false;
              Meldung( 'Kein Abschluss: Fachpraxis schlechter als 4' );
          end;
      end;

      if not Result then
      begin
    erfolg_ba := bwNichtBestanden;
    abschl_ba := '';
      Meldung( 'Ergebnis: Berufsabschlussprüfung nicht bestanden!' );
      end else
  begin
    erfolg_ba := bwBestanden;
    case FBKGLiederung of
      glB01: abschl_ba := POJahrErsetzen('APO-BK-15/B/BAB');
      glB08,glB09,glB10: abschl_ba := 'APO-BK-15/B08/BK/BAB';
    end;
      Meldung( 'Ergebnis: Berufsabschlussprüfung bestanden!' );
```

```
    end;

//Allgemein
        Meldung( ' ' );
        Meldung( 'Qualifikation ermitteln' );
        Meldung( '----------------------------------------');

// Eingangs-Quali prüfen
    case FBKGliederung of
    glB01:
      begin
        if FLSEntlassart = '' then
          Meldung( 'Keine Eingangsqualifikation gefunden' );

    // Prüfen ob    FOR + Q schon enthalten
        if FLSEntlassart = 'G' then
        begin
          Meldung( 'FOR + Q schon vorhanden' );
          erfolg_ab := bwIgnorieren;
          stat_krz := '5G';
          AbschlussErgebnisSpeichern( erfolg_ab, erfolg_ba, abschl_ab, abschl_ba, stat_krz );
          exit;     // schon enthalten
        end;

    //  Prüfen, ob FOR enthalten ist
        si_for := FLSEntlassart = 'F';

        si_10a := FLSEntlassart = 'D';

        qv := false;
        if si_for then
        begin
          qv := BK_QVermerkBerechnen;       // hat schon FOR, direkt Q-Vermerk berechnen
          if qv then
            quali := 'FOR + Q'
          else
            quali := 'FOR';
        end else if si_10a then
        begin       // Eingangsquali := SI 10A
          quali := 'SI 10A';
    // Feststellen, ob M oder E Grund- oder Erweiterungskurs ist (nur für Fachklasse
    Kinderpfleger)
          qv := BK_B_KinderpflegerEKurs( fJahr - 1, false );
          if qv then
            qv := BK_QVermerkBerechnen;
          if qv then
            quali := 'FOR + Q'
          else
            quali := 'FOR';
        end;
        if si_for then
          Meldung( 'Eingangsqualifikation: FOR' )
        else if si_10a then
          Meldung( 'Eingangsqualifikation: HSA' );
        Meldung( 'Erreichte Qualifikation: ' + quali );
        if quali = 'FOR + Q' then
        begin
          abschl_ab := POJahrErsetzen('APO-BK-15/B/FGO');
          stat_krz := '5G';
        end else if quali = 'FOR' then
        begin
          abschl_ab := POJahrErsetzen('APO-BK-15/B/FOR7');
          stat_krz := '5F';
        end else if quali = 'SI 10A' then
        begin
          abschl_ab := POJahrErsetzen('APO-BK-15/SI');
          stat_krz := '5A';
        end;
```

```
      end;

    glB08, glB09, glB10:
      begin
// Prüfen auf HSA
        quali := 'HSA';;
// Anzahl Noten = 6 prüfen
        idf := NoteVorhanden( fC0, 'Note', '=', 6, -1 );
        if AnzahlElemente( idf ) > 0 then
        begin // ein oder mehr 6
          for i := 1 to AnzahlElemente( idf) do
            StrAdd( SchreibeFachNote( fC0, StrToInt( EinzelElement( idf, i ) ) ), msg );
          quali := '';
        end else
        begin
      // Anzahl Noten = 5 prüfen
          idf := NoteVorhanden( fC0, 'Note', '=', 5, -1 );
          if AnzahlElemente( idf  ) > 1 then
          begin // zwei oder mehr 5
            for i := 1 to AnzahlElemente( idf ) do
              StrAdd( SchreibeFachNote( fC0, StrToInt( EinzelElement( idf, i ) ) ), msg );
            quali := '';;
          end;
        end;

        if quali = 'HSA' then
        begin // pRüfen, ob FOR
          erfolg_ab := bwBestanden;
// Deutsch , Mathe und Englisch in fC1
          Uebertragen( fC0, fC1, HauptfachID( fC0, 'Fach', 'D' ) );
          Uebertragen( fC0, fC1, HauptfachID( fC0, 'Fach', 'M' ) );
          Uebertragen( fC0, fC1, HauptfachID( fC0, 'Fach', 'E' ) );

          dn := 0;
          fC1.First;
          while not fC1.EOF do
          begin
            dn := dn + fC1.FieldByname( 'Note' ).AsFloat;
            fC1.Next;
          end;

          dn := ( int( dn/3*10 ) )*0.1; // auf eine Nachkommastalle, nicht runden

          if dn <= 3 then
          begin
            qv := BK_QVermerkBerechnen;     // hat schon FOR, direkt Q-Vermerk berechnen
            if qv then
            begin  // FORQ
              quali := 'FOR + Q';
              abschl_ab := 'APO-BK-15/B08/FGO';
              if erfolg_ba = bwBestanden then
                stat_krz := '5G'
              else
                stat_krz := '0G';
            end else
            begin
              quali := 'FOR';
              abschl_ab := 'APO-BK-15/B08/FOR';
              if erfolg_ba = bwBestanden then
                stat_krz := '5F'
              else
                stat_krz := '0F';
            end;
          end else
          begin  // nur HSA
            abschl_ab := 'APO-BK-15/B08/HSA';
            if erfolg_ba = bwBestanden then
              stat_krz := '5D'
```

```
            else
              stat_krz := '0D';
          end;
          Meldung( 'Erreichte Qualifikation: ' + quali );
        end else
        begin // kein HSA, prüfen, ob berufs-best
          erfolg_ab := bwNichtBestanden;
          abschl_ab := '';
          if erfolg_ba = bwBestanden then
            stat_krz := '5A'
          else
            stat_krz := '0A';
          Meldung( 'Keine Qualifikation' );
        end;
      end;
    end;

    AbschlussErgebnisSpeichern( erfolg_ab, erfolg_ba, abschl_ab, abschl_ba, stat_krz );

    end;

var
  idHF: string;

begin
    if fBK_Modus = '?' then
    begin
        Meldung( 'Die Prüfungsordnung des Schülers/der Schülerin umfasst eine
        Abschlussprüfung. Bitte verwenden Sie die Kartei-Seite "BK-Abschluss".' );
        exit;
    end;

    Result := 'N';
    msg := '';
// Schriftliche Fächer

    if fBK_Modus[1] in ['M','A'] then
    begin
        if FeldWertAnzahl( fC0, 'IstSchriftlich', '+' ) < 2 then
        begin
            Meldung( 'Falsche Anzahl schriftlicher Fächer' );
            exit;
        end;
    end;

// Deutsch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'D' );
  if idHF = '' then
    begin
        Meldung( 'Deutsch fehlt' );
        exit;
    end;

// Mathe prüfen
  idHF := HauptfachID( fC0, 'Fach', 'M' );
    if FeldWertAnzahl( fC0, 'Fach', 'M' ) = 0 then
    begin
        Meldung( 'Mathematik fehlt' );
        exit;
    end;

// Englisch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'E' );
  if idHF = '' then
    begin
        Meldung( 'Englisch fehlt' );
        exit;
    end;
```

```
// Fachpraxis prüfen
{   if FeldWertAnzahl( fC0, 'FachIntern', 'FP' ) = 0 then
    begin
        Meldung( 'Fachpraxis fehlt' );
        exit;
    end;}

// B01
    case fBK_Modus[1] of
    'Z' : // Zulassung prüfen
                if BK_B_ZulassungPruefen then
                    Result := fBK_Modus
                else
                    Result := 'N';
    'M' :   // Mündliche Noten festlegen
                begin
                    ResetBKMdlPruefungen;
                    MuendlichePruefungen;
                end;
    'A' : // Abschluss
                if AbschlussBerechnen then
                    Result := 'A';
    end;

end;

function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_B_BAB_Abschluss_B02_B07: string;
var
    idHF, quali, msg, idf: string;
    qv: boolean;
  erfolg_ab, erfolg_ba: TBestehungswert;
  abschl_ab, abschl_ba, stat_krz: string;

begin
// Deutsch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'D' );
  if idHF = '' then
    begin
        Meldung( 'Deutsch fehlt' );
        exit;
    end;

// Mathe prüfen
  idHF := HauptfachID( fC0, 'Fach', 'M' );
  if idHF = '' then
    begin
        Meldung( 'Mathematik fehlt' );
        exit;
    end;

// Englisch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'E' );
  if idHF = '' then
    begin
        Meldung( 'Englisch fehlt' );
        exit;
    end;

// Fachpraxis prüfen (nur bei Fachklassen 115 00 Sozial- und Gesundheitswesen)
    if ( FFachklasse = '11500' ) and ( FeldWertAnzahl( fC0, 'Fach', 'FP' ) = 0 ) then
    begin
        Meldung( 'Fachpraxis fehlt' );
        exit;
    end;

// B02, B07
    if not BK_B_ZulassungPruefen then
```

```
    begin
    erfolg_ab := bwNichtBestanden;
    abschl_ab := '';

    erfolg_ba := bwNichtBestanden;
    abschl_ba := '';
    stat_krz := '0A';
    end else
    begin
        FBildungsgang := 'B';
        fTeilPruefOrdKrz := POJahrErsetzen('APO-BK-15/B/BG');//'APO-BK-03/B/BG';
        AbschlussSpeichern( 1 );
// Allgemein-bildend
        FBildungsgang := 'A';
        qv := BK_QVermerkBerechnen;
        if qv then
            quali := 'FOR + Q'
        else
            quali := 'FOR';
        Meldung( 'Erreichte Qualifikation: ' + quali );

    case FBKGliederung of
    glB02:
      begin
        if quali = 'FOR + Q' then
        begin
          erfolg_ab := bwBestanden;
          abschl_ab := 'APO-BK-15/B/FGO';

          erfolg_ba := bwBestanden;
          abschl_ba := 'APO-BK-15/B/BG';
          stat_krz := '2G';
        end else if quali = 'FOR' then
        begin
          erfolg_ab := bwBestanden;
          abschl_ab := 'APO-BK-15/B/FOR7';

          erfolg_ba := bwBestanden;
          abschl_ba := 'APO-BK-15/B/BG';
          stat_krz := '2F';
        end;
      end;
    glB07:
      begin
        if quali = 'FOR + Q' then
        begin
          erfolg_ab := bwBestanden;
          abschl_ab := 'APO-BK-15/B/FGO';

          erfolg_ba := bwBestanden;
          abschl_ba := 'APO-BK-15/B/BK';
          stat_krz := '4G';
        end else if quali = 'FOR' then
        begin
          erfolg_ab := bwBestanden;
          abschl_ab := 'APO-BK-15/B/FOR';

          erfolg_ba := bwBestanden;
          abschl_ba := 'APO-BK-15/B/BK';
          stat_krz := '4F';
        end;
      end;
    end;
    end;
  AbschlussErgebnisSpeichern( erfolg_ab, erfolg_ba, abschl_ab, abschl_ba, stat_krz );
end;

function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_B_BAB_Abschluss_B03: string;
```

```
var
    idHF, msg, idf: string;
    qv: boolean;
begin
// Deutsch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'D' );
  if idHF = '' then
    begin
        Meldung( 'Deutsch fehlt' );
        exit;
    end;

// Mathe prüfen
  idHF := HauptfachID( fC0, 'Fach', 'M' );
  if idHF = '' then
    begin
        Meldung( 'Mathematik fehlt' );
        exit;
    end;

// Englisch prüfen
  idHF := HauptfachID( fC0, 'Fach', 'E' );
  if idHF = '' then
    begin
        Meldung( 'Englisch fehlt' );
        exit;
    end;

// Fachpraxis prüfen
{   if FeldWertAnzahl( fC0, 'FachIntern', 'FP' ) = 0 then
    begin
        Meldung( 'Fachpraxis fehlt' );
        exit;
    end;}

// B03
    if not BK_B_ZulassungPruefen then
    begin
        FBildungsgang := 'B';
        FAbgangsart := '0A';
        AbschlussSpeichern( 2 );
        FBildungsgang := 'A';
        AbschlussSpeichern( 2 );
    end else
    begin
        FBildungsgang := 'B';
        fTeilPruefOrdKrz := POJahrErsetzen('APO-BK-15/B/BG');//'APO-BK-03/B/BG';
        AbschlussSpeichern( 1 );
// Allgemein-bildend
        FBildungsgang := 'A';
        qv := BK_QVermerkBerechnen;
        if qv then
        begin
            Meldung( 'Q-Vermerk erreicht' );
            fTeilPruefOrdKrz := POJahrErsetzen('APO-BK-15/B/FGO');//'APO-BK-03/B/FGO';
            FAbgangsart := '2G';
        end else
        begin
            Meldung( 'Q-Vermerk nicht erreicht' );
            fTeilPruefOrdKrz := '';//'APO-BK-03/B/FOR6';        // oder FOR7???
            FAbgangsart := '2A';
        end;
        AbschlussSpeichern( 1 );
    end;

end;
```

```
// C01

function TBKPruefungsAlgorithmus.Pruefe_BK_C_Versetzung: string;
var
    idf, msg: string;
    i : integer;

begin
    Result := 'N';
    msg := '';

// Differenzierungsbereich raus
    AusContainerLoeschen( fC0, FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 ) );

// Auf 6 in C0 prüfen
    idf := NoteVorhanden( fC0, 'Note', '=', 6, -1 );
    if idf <> '' then
    begin       // 6 vorhanden
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        VersetzungsMeldung( 'Keine Versetzung:', msg );
        exit;
    end;

// Auf 5 in C0 prüfen
    idf := NoteVorhanden( fC0, 'Note', '=', 5, -1 );
    if NumToken( idf, ';' ) > 1 then
    begin   // mindestens eine 5 vorhanden
        for i := 1 to NumToken( idf, ';' ) do
             StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        VersetzungsMeldung( 'Keine Versetzung:', msg );
        exit;
    end;

    Result := 'V';
    Meldung( 'Versetzt!' );

end;


// C03
procedure TBKPruefungsAlgorithmus.Pruefe_BeruflicheKenntnisse_C03;
var
    msg, idf: string;
    i : integer;
    bk: boolean;
begin
// Die berufsbezogenen Fächer ermitteln
    msg := '';
    Uebertragen( fC0, fC1, FeldWertIDs( fC0, 'Fachgruppe_ID', '20', 0 ) );      // von C0
    nach C1

    bk := true;
    idf := NoteVorhanden( fC1, 'Note', '=', 6, -1 );
    if idf = '' then    // keine 6
    begin
        idf := NoteVorhanden( fC1, 'Note', '=', 5, -1 );
        if NumToken( idf , ';' ) > 1 then  // ein oder mehr 5
            bk := false;
    end else
        bk := false;

    FBildungsgang := 'B';
    if not bk then
    begin       // keine Beruflichen Kenntnisse
        for i := 1 to NumToken( idf, ';' ) do
            StrAdd( SchreibeFachNote( fC1, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        VersetzungsMeldung( 'Kein Zertifikat berufliche Kenntnisse:', msg );
```

```
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerLernabschnittsdaten SET
        AbschlussArt=%d, Abschluss=NULL WHERE ID=%d', [ 0, fA_ID] ) );
        SchuelerAbschlussSpeichern( '0A' );
        FAbgangsart := '0A';
    end else
    begin
        Meldung ( 'Mit Zertifikat berufliche Kenntnisse!' );
        fTeilPruefOrdKrz := POJahrErsetzen('APO-BK-15/C-BFS-BK/BK');//'APO-BK-03/C-BFS-BK/BK';
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerLernabschnittsdaten SET
        AbschlussArt=%d, Abschluss_B=%s WHERE ID=%d', [ 0, QuotedStr( fTeilPruefOrdKrz ),
        fA_ID] ) );
        FAbgangsart := '4A';
    end;
    SchuelerAbschlussSpeichern( FAbgangsart );
end;

function TBKPruefungsAlgorithmus.Zulassung_FHR_BAP( const zulassungsmodus: string ): boolean;
var
  i: integer;
  idf: string;
  cmd, msg, diff_ber: string;
begin
// HIer das Original aus C01, C02
{Anlage C §6 (2)  Zur  Prüfung  zum  Erwerb  der  Fachhochschulreife  wird  zugelas-
sen, wer in allen Fächern mindestens die Vornote „ausreichend“ oder
in nicht mehr als zwei Fächern die Vornote „mangelhaft“ erreicht hat.
Die Noten in abgeschlossenen Fächern werden einbezogen. Im Falle
einer ungenügenden Leistung ist eine Zulassung ausgeschlossen.}
// Laut Herrrn Pfotenhauer spiel der Diff-Bereich hierbei keine Rolle
// Differenzierungsbereich raus (nur temporär)

  diff_ber := FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 );
  Uebertragen( fC0, fC1, diff_ber );        // von C0 nach C1

  msg := '';
  Result := true;

  idf := NoteVorhandenGewichtung( fC0, 'Note', '=', 6, '' );
  if idf <> '' then// Anzahl Noten = 6 > 0?
    Result := false
  else
  begin
    idf := NoteVorhandenGewichtung( fC0, 'Note', '=', 5, '' );
    Result := NumToken( idf , ';' ) <= 2;  // zwei oder mehr 5
  end;

  if not Result then
  begin
    Meldung( Format( 'Zur %s-Prüfung nicht zugelassen', [ zulassungsmodus ] ) );
    for i := 1 to NumToken( idf, ';' ) do
      StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
  end else
  begin
    Meldung( Format( 'Zur %s-Prüfung zugelassen', [ zulassungsmodus ] ) );
  end;
  Uebertragen( fC1, fC0, diff_ber );        // Diff wieder zurück

end;

function TBKPruefungsAlgorithmus.Zulassung_ErweiterteBeruflicheKenntnisse: boolean;
var
  i: integer;
  msg, idf, res_erw: string;
begin
// Jetzt die Zulassung zu erw. berufl. Kenntnisse für C03, C04
  msg := '';
  Result := true;
// Die berufsbezogenen Fächer ermitteln
```

```
  Uebertragen( fC0, fC1, FeldWertIDs( fC0, 'Fachgruppe_ID', '20', 0 ) );        // von C0
  nach C1
// Auf 6 in C1 prüfen
  idf :=  NoteVorhandenGewichtung( fC1, 'Note', '=', 6, '' );
  if idf = '' then
  begin     // keine 6
    idf :=  NoteVorhandenGewichtung( fC1, 'Note', '=', 5, '' );
    Result := NumToken( idf, ';' ) <= 2;
  end else
    Result := false;

  if Result then
  begin
    Meldung( 'Zur Prüfung erweiterte berufl. Kenntnisse zugelassen' );
    res_erw := '+';
  end else
  begin
    for i := 1 to NumToken( idf, ';' ) do
      StrAdd( SchreibeFachNote( fC1, StrToInt( GetToken( idf, ';', i ) ) ), msg );
    VersetzungsMeldung( 'Zu Prüfung erweiterte berufl. Kenntnisse nicht zugelassen:', msg );
    res_erw := '-';
  end;
  TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET ZulassungErwBK=%s
  WHERE Schueler_ID=%d',
                           [ QuotedStr( res_erw ), fS_ID ] ) );
end;

function  TBKPruefungsAlgorithmus.AktuellerAbschlussIstHoeherAlsEingangsqualifikation( const
ab_abschl: string ): boolean;
var
  ix: integer;
  niedriger: string;
begin
  ix := FAbschlussHierarchie.IndexOfName( ab_abschl );
  if ix < 0 then
    Result := ab_abschl > FLSEntlassart // NEU ">" statt ">="
  else
  begin
    niedriger := FAbschlussHierarchie.ValueFromIndex[ ix ]; // Die Liste der niedigeren
    Abschlüsse
    Result := AnsiContainsText( niedriger, '-' + FLSEntlassart + '-' );
  end;
end;



procedure TBKPruefungsAlgorithmus.AbschlussErgebnisSpeichern( const bestanden_ab,
bestanden_bb: TBestehungswert; art_intern_ab, art_intern_bb, statkrz: string;
                                                              const meldung_text: string =
                                                              ''  );
var
  cmd1, cmd2, skrz: string;
  abschl_art: integer;
  ist_hoeher: boolean;
begin
  ist_hoeher := true;
  skrz := statkrz;
// das Statistik-Kürzel bei den Schülern eintragen
  if skrz <> '' then
  begin
// NEU: Prüfen, ob der allgemein-bildende Teil" des Statistk-Abschlusses >= vorhandener
höchster allg.buldender Abschluss ist
    if ( FLSEntlassart <> '' ) and ( length( skrz ) > 1 ) then
      ist_hoeher := AktuellerAbschlussIstHoeherAlsEingangsqualifikation( copy( skrz, 2, 1 ) );

    if not ist_hoeher and ( length ( skrz ) > 1 ) then
      skrz[2] := 'A'; // NEUES Verfahren, Juni 2015
```

```
//    if FBKGliederung = glA01 then   // NEU : Sonderregelug A01
//      skrz := '3A';

    cmd1 := Format( 'UPDATE Schueler SET EntlassArt=%s, Entlassjahrgang_ID=%d WHERE ID=%d',
    [ QuotedStr( skrz ), FJahrgang_ID, fS_ID] );
    TransactionHandler.DoExecute( cmd1 );
  end;

// Ist überhaupt ein Abschluss vorhanden?
  if ( bestanden_ab = bwNichtBestanden ) and ( bestanden_bb = bwNichtBestanden ) then
  begin // nicht bestanden
    cmd1 := Format( 'UPDATE SchuelerLernabschnittsdaten SET Abschlussart=2, Abschluss=NULL,
    Abschluss_B=NULL WHERE ID=%d',
                    [ fA_ID ] );
    cmd2 := Format( 'UPDATE SchuelerBKAbschluss SET Bestanden=%s, BestandenBA=%s WHERE
    Schueler_ID=%d',
                      [ QuotedStr( '-' ), QuotedStr( '-' ), fS_ID ] );
    TransactionHandler.DoExecute( cmd1 );
    TransactionHandler.DoExecute( cmd2 );
    exit;
  end else if ( bestanden_ab = bwBestanden ) or ( bestanden_bb = bwBestanden ) then
  begin
    cmd1 := Format( 'UPDATE SchuelerLernabschnittsdaten SET Abschlussart=1 WHERE ID=%d',
                   [ fA_ID ] );
    TransactionHandler.DoExecute( cmd1 );
{    if ( bestanden_ab = bwBestanden ) and ( bestanden_bb = bwBestanden ) then
      cmd2 := Format( 'UPDATE SchuelerBKAbschluss SET Bestanden=%s, BestandenBA=%s WHERE
      Schueler_ID=%d',
                        [ QuotedStr( '+' ), QuotedStr( '+' ), fS_ID ] )
    else if ( bestanden_ab = bwBestanden ) then
      cmd2 := Format( 'UPDATE SchuelerBKAbschluss SET Bestanden=%s WHERE Schueler_ID=%d',
                        [ QuotedStr( '+' ), fS_ID ] )
    else if ( bestanden_bb = bwBestanden )  then
      cmd2 := Format( 'UPDATE SchuelerBKAbschluss SET BestandenBA=%s WHERE Schueler_ID=%d',
                        [ QuotedStr( '+' ), fS_ID ] );
    TransactionHandler.DoExecute( cmd2 );}
  end;

// allgemein-bildend
  case bestanden_ab of
  bwBestanden:
    begin
      cmd2 := Format( 'UPDATE SchuelerBKAbschluss SET Bestanden=%s WHERE Schueler_ID=%d',
                     [ QuotedStr( '+' ), fS_ID ] );
    end;
  bwNichtBestanden:
    begin
      cmd2 := Format( 'UPDATE SchuelerBKAbschluss SET Bestanden=%s WHERE Schueler_ID=%d',
                     [ QuotedStr( '-' ), fS_ID ] );
    end;
  end;

  if cmd2 <> '' then
    TransactionHandler.DoExecute( cmd2 );

// Statistik-Kürzel für Abschluss setzen
  if ist_hoeher then
  begin
    cmd1 := '';
    cmd2 := '';
    case bestanden_ab of
    bwBestanden:
      begin
        cmd1 := Format( 'UPDATE SchuelerLernabschnittsdaten SET Abschluss=%s WHERE ID=%d',
                       [ QuotedStr( art_intern_ab ), fA_ID ] );
      end;
    bwNichtBestanden:
      begin
```

```
        cmd1 := Format( 'UPDATE SchuelerLernabschnittsdaten SET Abschluss=NULL WHERE ID=%d',
                        [ fA_ID ] );
      end;
    end;
    if cmd1 <> '' then
      TransactionHandler.DoExecute( cmd1 );
    if meldung_text <> '' then
      Meldung( meldung_text );
  end else
  begin
    if meldung_text <> '' then
      Meldung( 'Höherer allgemein-bildender Abschluss schon vorhanden' );
  end;

// Berufsbezogen
  cmd1 := '';
  cmd2 := '';
  case bestanden_bb of
  bwBestanden:
    begin
      cmd1 := Format( 'UPDATE SchuelerLernabschnittsdaten SET Abschluss_B=%s WHERE ID=%d',
                      [ QuotedStr( art_intern_bb ), fA_ID ] );
      cmd2 := Format( 'UPDATE SchuelerBKAbschluss SET BestandenBA=%s WHERE Schueler_ID=%d',
                      [ QuotedStr( '+' ), fS_ID ] );
    end;
  bwNichtBestanden:
    begin
      cmd1 := Format( 'UPDATE SchuelerLernabschnittsdaten SET Abschluss_B=NULL WHERE ID=%d',
                      [ fA_ID ] );
      cmd2 := Format( 'UPDATE SchuelerBKAbschluss SET BestandenBA=%s WHERE Schueler_ID=%d',
                      [ QuotedStr( '-' ), fS_ID ] );
    end;
  end;
  if cmd1 <> '' then
    TransactionHandler.DoExecute( cmd1 );
  if cmd2 <> '' then
    TransactionHandler.DoExecute( cmd2 );
end;

function TBKPruefungsAlgorithmus.Abschlussberechnen_ErweiterteBeruflicheKenntnisse:
TBestehungswert;
var
  i: integer;
  idf, msg: string;
begin
// Nun noch die erweiterten beruflichen Kenntnisse
// Die berufsbezogenen Fächer ermitteln
  msg := '';
  Uebertragen( fC0, fC1, FeldWertIDs( fC0, 'Fachgruppe_ID', '20', 0 ) );        // von C0
  nach C1
  idf := NoteVorhandenGewichtung( fC1, 'NoteAbschl', '>=', 5, 'Gewichtung' );
  if AnzahlElemente( idf ) > 1 then
  begin     // Keine Erweiterten Kenntnisse
    for i := 1 to NumToken( idf, ';' ) do
      StrAdd( SchreibeFachNote( fC1, StrToInt( EinzelElement( idf, i ) ) ), msg );
    VersetzungsMeldung( 'Keine erweiterten beruflichen Kenntnisse: ', msg );
    TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET BestandenErwBK=%s
    WHERE Schueler_ID=%d',
                                [ QuotedStr( '-' ), fS_ID ] ) );
  end else
  begin // Erweiterte Kenntnisse
    Meldung( 'Erweiterte berufliche Kenntnisse erlangt' );
    TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET BestandenErwBK=%s
    WHERE Schueler_ID=%d',
                                [ QuotedStr( '+' ), fS_ID ] ) );
  end;
end;
```

```
function TBKPruefungsAlgorithmus.Abschlussberechnen_BA: TBestehungswert;
var
  sum: double;
  i, id, cnt: integer;
  idf: string;
  msg: string;
begin
  msg := '';
  if IstEGliederung then
  begin
// Torspecken: Der Fachschulabschluss wird nur vergeben, wenn die Leistungen in den 3
Abschlussarneiten im Schnitt 4,0
//oder besser sind.
    idf := FeldWertIDs( fC0, 'Fachgruppe_ID', IntToStr( C_FG_ABSCHLUSSARBEIT ), 0 );
    if AnzahlElemente( idf ) < 3 then
    begin
      Result := bwNichtBestanden;
      msg := 'Zu wenig Abschlussarbeiten gefunden';
      VersetzungsMeldung( 'Kein Berufsabschluss: ', msg );
      exit;
    end;
    sum := 0;
    cnt := 0;
    for i := 1 to AnzahlElemente( idf ) do
    begin
      id := StrToInt( EinzelElement( idf, i ) );
      if fC0.Locate( 'ID', id, [] ) then
      begin
        if FC0.FieldByname( 'NoteAbschl' ).AsInteger <> 999  then
        begin
          sum := sum + FC0.FieldByname( 'NoteAbschl' ).AsInteger;
          inc( cnt );
        end;
      end;
    end;
    sum := sum / cnt;
    if cnt < 3 then
    begin
      Result := bwNichtBestanden;
      msg := 'Facharbeiten nicht korrekt benotet';
      VersetzungsMeldung( 'Kein Berufsabschluss: ', msg );
    end else if ( sum > 4 ) then
    begin
      Result := bwNichtBestanden;
      msg := 'Durchschnittsnote der Facharbeiten nicht ausreichend';
      VersetzungsMeldung( 'Kein Berufsabschluss: ', msg );
    end else
    begin
      Result := bwBestanden;
      Meldung( 'Berufsabschluss erreicht' );
    end;

  end else
  begin
// Ab hier wieder für C01 und C02
// Beruflicher Abschluss
    Result := bwBestanden;
    idf := NoteVorhandenGewichtung( fC0, 'NoteAbschl', '=', 6, 'Gewichtung' );

    if idf <> '' then  // mindestens eine 6
    begin
      for i := 1 to NumToken( idf, ';' ) do
        StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
      Result := bwNichtBestanden;
    end else
    begin       // keine 6
      idf := NoteVorhandenGewichtung( fC0, 'NoteAbschlBA', '=', 5, 'Gewichtung' );  //
```

```
      Prüfen, wieviel 5
      if NumToken( idf, ';' ) > 1 then
      begin     // mehr als eine 5
        for i := 1 to NumToken( idf, ';' ) do
          StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
        Result := bwNichtBestanden;
      end else if ( FPraktPrfNote = 999 ) then
      begin
        StrAdd( 'Note Praktische Prüfung fehlt', msg );
        Result := bwNichtBestanden;
      end else if ( FPraktPrfNote > 4 ) then
      begin
        StrAdd( 'Praktische Prüfung nicht bestanden', msg );
        Result := bwNichtBestanden;
      end;
    end;

    if Result = bwBestanden then
      Meldung( 'Berufsabschluss erreicht' )
    else
      VersetzungsMeldung( 'Kein Berufsabschluss: ', msg );
  end;
end;


function TBKPruefungsAlgorithmus.AbschlussBerechnen_FHR: TBestehungswert;
var
  anz_schr, i: integer;
  fehler: boolean;
  idf, idfa, msg, diff_ber: string;
  DSN: double;
begin

  diff_ber := FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 );
  Uebertragen( fC0, fC1, diff_ber );        // von C0 nach C1

  Result := bwIgnorieren;
// Prüfen, ob genügend schriftliche Fächer
  fehler := false;
  anz_schr := FeldWertAnzahl( fC0, 'IstSchriftlich', '+' );
  case FBKGliederung of
  glA02:
    fehler := anz_schr <> 3;
  glC01:
    if IstSpezielleFachklasse( '10-114-00;100-114-00' ) then
      fehler := anz_schr <> 2
    else
      fehler := anz_schr <> 3;
  glC02:
    fehler := anz_schr <> 3;
  glC03, glC06, glC07, glC08:
    fehler := anz_schr <> 4;
  glE01..glE13:
    fehler := anz_schr > 1;
  end;
  if fehler then
  begin
    if FBKGliederung in [ glE01..glE13 ] then
    begin
        Meldung( 'Zu wenige schriftliche Fächer für FHR-Prüfung' );
      exit;
    end else
        Meldung( 'Hinweis: Falsche Anzahl schriftlicher Fächer für FHR-Prüfung?' );
//    Meldung( 'Falsche Anzahl schriftlicher Fächer' );
//    exit;
  end;

//  if not ( FBKGliederung in [ glE01..glE13 ] ) then
```

```
  begin
    fehler := FeldWertAnzahl( fC0, 'NoteSchr', '999' ) > 0;
    if fehler then
    begin
      Meldung( 'Schriftliche Noten unvollständig' );
      exit;
    end;
  end;

  fehler := FeldWertAnzahl( fC0, 'NoteMdl', '999' ) > 0;
  if fehler then
  begin
    Meldung( 'Mündliche Noten unvollständig' );
    exit;
  end;

  fehler := FeldWertAnzahl( fC0, 'NoteAbschl', '999' ) > 0;
  if fehler then
  begin
    Meldung( 'Abschlussnoten unvollständig' );
    exit;
  end;

  if not IstEGliederung then
  begin
    Meldung ( ' ' );
    Meldung( 'Abschluss-Berechnung' );
    Meldung( '-----------------------------------------');
  end;

// prüfen, ob FHR erreicht wurde, nur bei C01 und A02
{Anlage C, §12 (4) Die Prüfung ist bestanden, wenn in allen Fächern mindestens
ausreichende  Leistungen  erzielt  werden  oder  wenn  die  Leistungen
nur in einem Fach „mangelhaft“ sind und durch eine mindestens be-
friedigende Leistung in einem anderen Fach ausgeglichen werden.}
  if IstEGliederung  then
  begin
    if anz_schr = 0 then
    begin
      Result := bwNichtBestanden;
      msg := 'Kein schriftliches Fach';
    end else
    begin
//Die FHR-Prüfung ist bestanden, bei ausreichender Abschlussnote im FHR-Fach und bestandenem
Fachschulexamen.
      idf := FeldWertIDs( fC0, 'IstSchriftlich', '+', 0 );
      idfa := NoteVorhanden( fC0, 'NoteAbschl', '<=', 4, -1 );
      if not IstLeer( SchnittMenge( idf, idfa ) ) then
      begin
        Result := bwBestanden;
      end else
      begin
        Result := bwNichtBestanden;
        msg := 'Abschlussnote nicht ausreichend';
      end;
    end;
  end else
  begin
    idf := NoteVorhanden( fC0, 'NoteAbschl', '=', 6, -1 );
    if idf <> '' then
    begin       // eine oder mehr 6
      for i := 1 to AnzahlElemente( idf ) do
        StrAdd( SchreibeFachNoteFeld( fC0, StrToInt( EinzelElement( idf, i ) ), 'NoteAbschl'
        ), msg );
      Result := bwNichtBestanden;
    end else
    begin
      idf := NoteVorhanden( fC0, 'NoteAbschl', '=', 5, -1 );
```

```
      if AnzahlElemente( idf ) > 1 then
      begin // zwei oder mehr 5
        for i := 1 to AnzahlElemente( idf ) do
          StrAdd( SchreibeFachNoteFeld( fC0, StrToInt( EinzelElement( idf, i ) ),
          'NoteAbschl' ), msg );
        Result := bwNichtBestanden;
      end else if AnzahlElemente( idf  ) = 1 then
      begin     // eine 5, prüfen ob Fächer besser 3 vorhanden
        idfa := NoteVorhanden( fC0, 'NoteAbschl', '<=', 3, -1 );
        if idfa = '' then
        begin
          StrAdd( SchreibeFachNoteFeld( fC0, StrToInt( idf ), 'NoteAbschl' ), msg );
          msg := msg + ';' + 'Kein Ausgleichsfach gefunden';
          Result := bwNichtBestanden;
        end;
      end;
    end;
  end;

  if Result = bwNichtBestanden then
  begin     // kein FHR-Abschluss
    VersetzungsMeldung( 'Kein Abschluss FHR: ', msg );
    Meldung ( ' ' );
    TransactionHandler.DoExecute( Format( 'UPDATE Schueler SET Durchschnittsnote=NULL,
    DSN_Text=NULL WHERE ID=%d', [ fS_ID ] ) );
    TransactionHandler.DoExecute( Format( 'UPDATE Schueler SET DurchschnittsnoteFHR=NULL,
    DSN_FHR_Text=NULL WHERE ID=%d', [ fS_ID ] ) );
    exit;
  end;

// Nun die Durchschnittsnote aus den Abschlussnoten ermitteln
// NEU Jan. 2014, für FHR-DS den Diff-Bereich wieder rein.
  if diff_ber <> '' then
    Uebertragen( fC1, fC0, diff_ber );      // von C0 nach C1
// C1 leeren
  fC1.EmptyTable;
  DSN := BK_NotenDurchschnittFHR( fC0, 'NoteAbschl' );
  case FBKGliederung of
  glA02: DurchschnittsNoteSpeichern( DSN, 'DurchschnittsnoteFHR', 'DSN_FHR_Text' );
  else
    begin
      DurchschnittsNoteSpeichern( DSN, 'Durchschnittsnote', 'DSN_Text' );
      DurchschnittsNoteSpeichern( DSN, 'DurchschnittsnoteFHR', 'DSN_FHR_Text' );
    end;
  end;
  Meldung( Format( 'Abschluss FHR erreicht (Durchschnittsnote %s)', [ FloatToStr( DSN ) ] ) );
  Result := bwBestanden;
end;

function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_Abschluss_C: string;
var
  erfolg_fhr, erfolg_ba: TBestehungswert;
  cmd, msg, stat_krz, abschluss_ab, abschluss_bb: string;
  zugelassen: boolean;
  szugel: string;
begin
// C01  (Abschluss:  Berufsausbildung + FHR)    ->  C1, C3, C7
// C02  (Abschluss: Berufsausbildung, Voraussetzung FHR/Abi)    ->  C2, C4
// C03  (Abschluss: erweiterte berufliche Kenntnisse + FHR) ->  C5
// C04  (Abschluss: erweiterte berufliche Kenntnisse, Voraussetzung FHR/Abi)    ->  C6
// C05  (Klasse 11 Fachoberschule mit Jahrespraktikum)      ->  C9 (1. Jahr)
// C06  (Klasse 12 Fachoberschule führt zu vertieften berufliche Kenntnisse + FHR)  ->  C9
// C07  (Klasse 12 Fachoberschule mit Berufsausbildung als Voraussetzung führt zu vertieften
beruflichen Kenntnissen + FHR in Teilzeit – 2jährig)          ->  C11
// C08  (Klasse 12 Fachoberschule mit Berufsausbildung als Voraussetzung führt zu vertieften
beruflichen Kenntnissen + FHR in Vollzeit – 1jährig)          ->  C10

{Folgerung
```

```
C01: FHR + BA
C02: nur BA
C03: FHR- und erw. BK-Prüfung
C04: Nur erw. BK-Prüfung
C06: FHR
C07: FHR
C08: FHR}
  msg := '';

    case fBK_Modus[1] of
    '?' : Meldung( 'Die Prüfungsordnung des Schülers/der Schülerin umfasst eine
    Abschlussprüfung. Bitte verwenden Sie die Kartei-Seite "BK-Abschluss".' );
    'Z' : // Zulassung prüfen
    begin
      ResetBKAbschluss( fS_ID );
      TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKFaecher SET MdlPruefung=%s,
      MdlPruefungFW=%s, NoteMuendlich=NULL, NoteAbschluss=NULL WHERE Schueler_ID=%d',
                           [ QuotedStr( '-' ), QuotedStr( '-' ), fS_ID ] ) );
      Meldung( ' ' );
      Meldung( 'Ergebnis Zulassungsprüfung' );
      Meldung( '-------------------------' );
      case FBKGliederung of
        glA02, glC01: zugelassen := Zulassung_FHR_BAP( 'FHR' );
        glC02: zugelassen := Zulassung_FHR_BAP( 'BA' ); // FHR Voraussetzung
        glC03: zugelassen := Zulassung_FHR_BAP( 'FHR' );// and
        Zulassung_ErweiterteBeruflicheKenntnisse;
        glC04: zugelassen := true;//Zulassung_ErweiterteBeruflicheKenntnisse; // oder per se
        zugelassen?
        glC06: zugelassen := Zulassung_FHR_BAP( 'FHR' );
        glC07: zugelassen := Zulassung_FHR_BAP( 'FHR' );
        glC08: zugelassen := Zulassung_FHR_BAP( 'FHR' );
        glD05, glD06: zugelassen := Zulassung_FHR_BAP( 'Abitur' );
      end;
      if zugelassen then
      begin
        Result := fBK_Modus;
        szugel := '+';
      end else
      begin
        Result := 'N';
        szugel := '-';
      end;

      case FBKGliederung of
      glA02, glC01: // BA und FHR
        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s, ZulassungBA=%s WHERE
        Schueler_ID=%d',
                       [ QuotedStr( szugel ), QuotedStr( szugel ), fS_ID ] );
      glC02: // BA (FHR ist Vorauss.)
        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s, ZulassungBA=%s WHERE
        Schueler_ID=%d',
                       [  QuotedStr( szugel ), QuotedStr( szugel ),    fS_ID ] );
      glC03: // ebK (FHR ist Vorauss.)
//        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s, ZulassungErwBK=%s
WHERE Schueler_ID=%d',
//                       [ QuotedStr( szugel ), QuotedStr( szugel ), fS_ID ] );
// NEU 11/2013: Keine ErwBK
        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s WHERE Schueler_ID=%d',
                       [ QuotedStr( szugel ), fS_ID ] );
      glC04: // ebK (FHR ist Vorauss.)
//        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s, ZulassungErwBK=%s
WHERE Schueler_ID=%d',
//                       [ QuotedStr( szugel ), QuotedStr( szugel ), QuotedStr( szugel ),
fS_ID ] );
// NEU 11/2013: Keine ErwBK
        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s WHERE Schueler_ID=%d',
                       [ QuotedStr( szugel ), fS_ID ] );
      glC05: // nur FHR
```

```
        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s WHERE Schueler_ID=%d',
                       [ QuotedStr( szugel ), fS_ID ] );
      glC06: // nur FHR
        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s WHERE Schueler_ID=%d',
                       [ QuotedStr( szugel ), fS_ID ] );
      glC07: // nur FHR
        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s WHERE Schueler_ID=%d',
                       [ QuotedStr( szugel ), fS_ID ] );
      glC08: // nur FHR
        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s WHERE Schueler_ID=%d',
                       [ QuotedStr( szugel ), fS_ID ] );
      glD05, glD06: // Abitur
        cmd := Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s WHERE Schueler_ID=%d',
                       [ QuotedStr( szugel ), fS_ID ] );
      end;
      TransactionHandler.DoExecute( cmd );
    end;
    'A' : // Abschluss
    begin
//      if AbschlussBerechnen_C03_C04 then
//        Result := 'A';
      stat_krz := '';
      abschluss_ab := '';
      abschluss_bb := '';
      case FBKGliederung of
      glA02:
        begin
          erfolg_fhr := AbschlussBerechnen_FHR;
          erfolg_ba := bwIgnorieren;
          if erfolg_fhr = bwBestanden then
          begin
            stat_krz := '3J'; // Berufsabschluss und Fachhochschulreife
            abschluss_ab := POJahrErsetzen('APO-BK-15/A-FK/FHR');//'APO-BK-03/A-FK/FHR';
          end;
        end;

      glC01:
        begin
          erfolg_fhr := AbschlussBerechnen_FHR;
          erfolg_ba := AbschlussBerechnen_BA;
          if erfolg_fhr = bwBestanden then
            abschluss_ab :=
            POJahrErsetzen('APO-BK-15/C-BFS-BAB/FHR');//'APO-BK-03/C-BFS-BAB/FHR';
          if erfolg_ba = bwBestanden then
            abschluss_bb :=
            POJahrErsetzen('APO-BK-15/C-BFS-BAB/BAB');//'APO-BK-03/C-BFS-BAB/BAB';
          if ( erfolg_fhr = bwBestanden ) and ( erfolg_ba = bwBestanden ) then
            stat_krz := '5J' // Berufsabschluss un Fachhochschulreife
          else if erfolg_ba = bwBestanden then
            stat_krz := '5A' // Berufsabschluss
          else if erfolg_fhr = bwBestanden then
            stat_krz := '4H' // Fachhochschulreife schulischer Teil
          else
            stat_krz := '0A'; // ohne Abschluss
        end;

      glC02:
        begin
// Soll hier nochmal die FHR eingetragen werden (obwohl Eingangsvoraussetzung)???
          erfolg_fhr := bwIgnorieren;
          erfolg_ba := AbschlussBerechnen_BA;
          if erfolg_ba = bwBestanden then
          begin
            abschluss_bb :=
            POJahrErsetzen('APO-BK-15/C-BFS-BAB/BAB');//'APO-BK-03/C-BFS-BAB/BAB';
            stat_krz := '5A'; // Berufsabschluss
          end else
            stat_krz := '0A';
```

```
        end;

      glC03:
        begin
          erfolg_fhr := AbschlussBerechnen_FHR;
          erfolg_ba := bwIgnorieren;//AbschlussBerechnen_ErweiterteBeruflicheKenntnisse;
          if erfolg_fhr = bwBestanden then
            abschluss_ab := POJahrErsetzen( 'APO-BK-15/C-BFS-BK/FHR-S'
            );//'APO-BK-03/C-BFS-BK/FHR-S';
{           if erfolg_ba = bwBestanden then
            abschluss_bb := 'APO-BK-11/C-BFS-BK/EBK1';//'APO-BK-03/C-BFS-BK/EBK1';
          if ( erfolg_fhr = bwBestanden ) and ( erfolg_ba = bwBestanden ) then
            stat_krz := '7H' // erweiterte berufliche Kenntnisse und Fachhochschulreife
          else if erfolg_ba = bwBestanden then
            stat_krz := '7A' // erweiterte berufliche Kenntnisse
          else if erfolg_fhr = bwBestanden then
            stat_krz := '4H' // berufliche kenntnisse und Fachhochschulreife schulischer Teil
          else if erfolg_ba = bwNichtBestanden then
          begin
            stat_krz := '4A'; // berufliche Kenntnisse sind auf jeden Fall vorhanden
            abschluss_bb := 'APO-BK-11/C-BFS-BK/BK';//'APO-BK-03/C-BFS-BK/BK';
            erfolg_ba := bwBestanden; // damit die Daten übernommen werden
          end;}

// NEU: Nur noch FHR möfglich
         if erfolg_fhr = bwBestanden then
            stat_krz := '4H'; // NEU Juni 2016

        end;

      glC04:
        begin
// Soll ier nochmal die FHR eingetragen werden (obwohl Eingangsvoraussetzung)???
          erfolg_fhr := bwIgnorieren;
          erfolg_ba := bwIgnorieren;//AbschlussBerechnen_ErweiterteBeruflicheKenntnisse;
          if erfolg_ba = bwBestanden then
          begin
            abschluss_bb :=
            POJahrErsetzen('APO-BK-15/C-BFS-BK/EBK2');//'APO-BK-03/C-BFS-BK/EBK2';
            stat_krz := '7A'; // erweiterte berufliche Kenntnisse
          end else
            stat_krz := '0A'; // oder 4A??berufliche Kenntnisse sind auf jeden Fall vorhanden
        end;

      glC06, glC07, glC08:
        begin
          erfolg_fhr := AbschlussBerechnen_FHR;
          erfolg_ba := bwIgnorieren;
          if erfolg_fhr = bwBestanden then
          begin
            abschluss_ab := POJahrErsetzen('APO-BK-15/C-FOS/FHR');//'APO-BK-03/C-FOS/FHR';
            case FBKGliederung of
            glC06: stat_krz := '4J'; // Fachhochschulreife, Pfotenhauer: C06 bekommt jetzt 4J
            glC07, glC08: stat_krz := '8J'; // Vertiefte berufl.Kenntnisse und
            Fachhochschulreife
            end;
          end else if erfolg_fhr = bwNichtBestanden then
          begin
            case FBKGliederung of
            glC06: stat_krz := '0A';
            glC07, glC08: stat_krz := '8D'; // Vertiefte berufl.Kenntnisse
{Pfotenhauer: C07 und C08: 8A fällt weg dafür gibt es jetzt 8D,8G,8J
8D ist der "berufliche Kenntnisse mit HA10, es kann sein, das die Schüler den jetzt immer
erwerben.
8G und 8J sind FOR und FHR dazu müssten in Schild aber dann weitere Prüfungen stattfinden.}
            end;
          end;
        end;
```

```
      glD05, glD06:
// Neu 3.7.17: Berechnung auch für D05, D06
        begin
          erfolg_fhr := AbschlussBerechnen_FHR;
          erfolg_ba := bwIgnorieren;
          if erfolg_fhr = bwBestanden then
          begin
            abschluss_ab := POJahrErsetzen('APO-BK-15/D-FOS13' );//'APO-BK-03/C-FOS/FHR';
            if FAnzSprachenBK < 2 then
            begin
              stat_krz := '8Q';
                    Meldung( 'Kein Nachweis 2. Fremdsprache vorhanden, fachgebundene
                    Hochschulreife erreicht' );
            end else
            begin
              stat_krz := '8K';
                    Meldung( 'Nachweis 2. Fremdsprache vorhanden, allgemeine Hochschulreife
                    erreicht' );
            end;
          end else if erfolg_fhr = bwNichtBestanden then
          begin
            stat_krz := '0A';
          end;
        end;
      end;
      if stat_krz <> '' then
        AbschlussErgebnisSpeichern( erfolg_fhr, erfolg_ba, abschluss_ab, abschluss_bb,
        stat_krz );
      if ( erfolg_fhr = bwBestanden ) or ( erfolg_ba = bwBestanden ) then
        Result := 'A'
      else
        Result := 'N';
    end;
  end;

end;

/// E-Gliederung


function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_Abschluss_E: string;
var
  erfolg_fhr, erfolg_ba: TBestehungswert;
  cmd, msg, stat_krz, abschluss_ab, abschluss_bb: string;
  zugelassen: boolean;
  szugel: string;
begin
//APO-BK-11/E
//APO-BK-11/E/FOR
//APO-BK-11/E/FHR
//In Anlage E können nur die Abschlüsse
// 6A (Fachschulabschluss)
// 6F (Fachschulabschluss und Mittlerer Abschluss)
// und 6J (Fachschulabschluss und Fachhochschulreife) erreicht werden.
//Abhängig von der Vorbildung dürfte 6F für die Schüler ohne FHR-Erfolg i.a.korrekt sein.


  msg := '';

    case fBK_Modus[1] of
    '?' : Meldung( 'Die Prüfungsordnung des Schülers/der Schülerin umfasst eine
    Abschlussprüfung. Bitte verwenden Sie die Kartei-Seite "BK-Abschluss".' );
    'A' : // Abschluss
    begin
      Meldung ( ' ' );
      Meldung( 'Abschluss-Berechnung' );
      Meldung( '-----------------------------------------');
```

```
      stat_krz := '';
      abschluss_ab := '';
      abschluss_bb := '';
      erfolg_ba := AbschlussBerechnen_BA;
      if erfolg_ba = bwNichtBestanden then
        erfolg_fhr := bwNichtBestanden
      else
        erfolg_fhr := AbschlussBerechnen_FHR;

      if erfolg_fhr = bwBestanden then
      begin
        abschluss_ab := POJahrErsetzen('APO-BK-15/E/FHR');
        stat_krz := '6J'; //Fachschulabschluss und Fachhochschulreife
      end else
      begin
        abschluss_ab := POJahrErsetzen('APO-BK-15/E/FOR');
        stat_krz := '6F'; //Fachschulabschluss und Mittlerer Abschluss  // lt. Torspecken 6A
      end;
      if stat_krz <> '' then
        AbschlussErgebnisSpeichern( erfolg_fhr, erfolg_ba, abschluss_ab, abschluss_bb,
        stat_krz );
      if ( erfolg_fhr = bwBestanden ) or ( erfolg_ba = bwBestanden ) then
        Result := 'A'
      else
        Result := 'N';
    end;
  end;
end;


/////// D03 D04
function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_Abschluss_D03_D04: string;
var
    erf: boolean;
    GN : double;
begin
    ResetBKAbschluss( fS_ID );
    fTeilPruefOrdKrz := POJahrErsetzen('APO-BK-15/D-BAB/BP');//'APO-BK-03/D-BAB/BP';
    Result := 'N';
    if ( FPraktPrfNote = 999 ) then
    begin
        Meldung( 'Abschlussnote Berufspraktikum fehlt' );
        erf := false;
    end else if ( FPraktPrfNote > 4 ) then
    begin
        Meldung( 'Berufspraktikum nicht bestanden' );
        erf := false;
    end else
        erf := true;

    if erf then
    begin
        if FNoteKolloquium = 999 then
        begin
            Meldung( 'Note Kolloquium fehlt' );
            erf := false;
        end else
        begin       // Gesamtnote berechnen: ( 2*Berufspraktikum + Kolloquium ) / 3
            GN := ( 2*FPraktPrfNote + FNoteKolloquium ) / 3;
            GN := int( GN * 10 );
            GN := GN * 0.1;
            DurchschnittsnoteSpeichern( GN, 'Durchschnittsnote', 'DSN_Text' );
            erf := GN <= 4;
            if erf then
                Meldung( Format( '%s (Durchschnittsnote %s)', [ 'Qualifikation "Staatlich
                anerkannte/r Erzieher/in"', FloatToStr( GN ) ] ) )
            else
                Meldung( Format( '%s (Durchschnittsnote %s)', [ 'Keine Qualifikation
```

```
                "Staatlich anerkannte/r Erzieher/in"', FloatToStr( GN ) ] ) );
        end;
    end;


    if not erf then
    begin
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s,
        BestandenBA=%s WHERE Schueler_ID=%d',
                                                        [ QuotedStr( '-' ),
                                                        QuotedStr( '-' ), fS_ID ] ) );
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerLernabschnittsdaten SET
        AbschlussArt=%d, Abschluss=NULL WHERE ID=%d', [ 2, fA_ID] ) );
        TransactionHandler.DoExecute( Format( 'UPDATE Schueler SET EntlassArt=%s,
        Entlassjahrgang_ID=%d WHERE ID=%d', [ QuotedStr( '0A' ), FJahrgang_ID, fS_ID] ) );
    end else
    begin
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s,
        BestandenBA=%s WHERE Schueler_ID=%d',
                                                        [ QuotedStr( '+' ),
                                                        QuotedStr( '+' ), fS_ID ] ) );
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerLernabschnittsdaten SET
        AbschlussArt=%d, Abschluss=%s WHERE ID=%d', [ 1, QuotedStr( fTeilPruefOrdKrz ),
        fA_ID] ) );
// Folgendes laut Angabe von Herrn Krahn
        TransactionHandler.DoExecute( Format( 'UPDATE Schueler SET EntlassArt=%s,
        Entlassjahrgang_ID=%d WHERE ID=%d', [ QuotedStr( '5A' ), FJahrgang_ID, fS_ID] ) );
        Result := 'A';
    end;
end;

/////// D05 D06
function TBKPruefungsAlgorithmus.Pruefe_APO_BK_15_Abschluss_D05_D06: string;
var
    idf, msg: string;

    function Zugelassen_D05_D06: boolean;
    var
        i : integer;
        res: string;
    begin
// Differenzierungsbereich ist nicht relevant
        AusContainerLoeschen( fC0, FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 ) );

        ResetBKAbschluss( fS_ID );
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKFaecher SET MdlPruefung=%s,
        MdlPruefungFW=%s, NoteMuendlich=NULL, NoteAbschluss=NULL WHERE Schueler_ID=%d',
                                                    [ QuotedStr( '-' ), QuotedStr(
                                                    '-' ), fS_ID ] ) );

//      if fAnzLeer > 0 then
//          Meldung( 'Hinweis: Nicht alle Fächer benotet' );

// Auf 6 in C0 prüfen
        idf := NoteVorhanden( fC0, 'Note', '=', 6, -1 );
        if idf = '' then
        begin       // keine 6
            idf := NoteVorhanden( fC0, 'Note', '=', 5, -1 );
            Result := NumToken( idf, ';' ) <= 2;
        end else
            Result := false;

        if Result then
        begin
            Meldung( 'Zur Abitur-Prüfung zugelassen' );
            res := '+';
        end else
        begin
```

```
            for i := 1 to NumToken( idf, ';' ) do
                StrAdd( SchreibeFachNote( fC0, StrToInt( GetToken( idf, ';', i ) ) ), msg );
            VersetzungsMeldung( 'Zur Abitur-Prüfung nicht zugelassen:', msg );
            res := '-';
        end;
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET Zulassung=%s,
        ZulassungErwBK=NULL, Bestanden=NULL, BestandenErwBK=NULL WHERE Schueler_ID=%d',
                                                                [ QuotedStr( res ), fS_ID ]
                                                                ) );
    end;

    function MuendlichePruefungen_D05_D06: boolean;
    var
        cnt : integer;
    begin
        Result := true;
// Differenzierungsbereich ist nicht relevant
        AusContainerLoeschen( fC0, FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 ) );
// Schiftliche Fächer in fC1 übertragen
        Uebertragen( fC0, fC1, FeldWertIDs( fC0, 'IstSchriftlich', '+', 0 ) );
        if fC1.RecordCount <{>} 2 then
        begin
            Meldung( 'Falsche Anzahl schriftlicher Fächer' );
            Result := false;
            exit;
        end;
        if FeldWertAnzahl( fC1, 'NoteSchr', '999' ) > 0 then
        begin
            Meldung( 'Schriftliche Noten unvollständig' );
            Result := false;
            exit;
        end;

        fC1.First;
        cnt := 0;
        while not fC1.EOF do
        begin
            if ( ( fC1.FieldByname( 'Note' ).AsFloat = 5 ) and ( fC1.FieldByname( 'NoteSchr'
            ).AsFloat = 4 ) ) or
                 ( abs( fC1.FieldByname( 'Note' ).AsFloat - fC1.FieldByname( 'NoteSchr'
                 ).AsFloat ) >=2 ) then
            begin
                if msg <> '' then
                    msg := msg + ';';
                msg := msg + Format( '%s: Vornote=%s, Prüfungsnote=%s',
                                                        [ fC1.FieldByname( 'Fachname'
                                                        ).AsString,
                                                            fC1.FieldByname( 'Note'
                                                            ).AsString,
                                                            fC1.FieldByname( 'NoteSchr'
                                                            ).AsString ] );
                TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKFaecher SET
                MdlPruefung=%s WHERE Schueler_ID=%d AND Fach_ID=%d',
                                                            [ QuotedStr( '+' ), fS_ID,
                                                            fC1.FieldByname( 'Fach_ID'
                                                            ).AsInteger ] ) );
                inc( cnt );

            end;
            fC1.Next;
        end;

        Meldung ( ' ' );
        Meldung( 'Mündliche Prüfung' );
        Meldung( '---------------------------------' );
        if cnt = 0 then
            Meldung( 'Keine mündlichen Prüfungen notwendig' )
        else
```

```
            VersetzungsMeldung( 'Abweichung Vornote/Prüfungsnote:' , msg );
    end;        // end function

    function AbschlussBerechnen_D05_D06: boolean;
    var
        idf, idfa: string;
        i : integer;
    inote: integer;
        an, DSN: double;
    nprfg: string;
    begin
        if IstInListe( fTeilPruefOrdKrz,
        'APO-BK-03/D-FOS13/FGHR;APO-BK-11/D-FOS13/FGHR;APO-BK-15/D-FOS13/FGHR' ) then
        begin       // Gesamter Alg. braucht nur für diese PO durchlaufen werden
            if FeldWertAnzahl( fC0, 'IstSchriftlich', '+' ) < 2 then
            begin
                Meldung( 'Falsche Anzahl schriftlicher Fächer' );
                exit;
            end;
            if FeldWertAnzahl( fC0, 'NoteSchr', '999' ) > 0 then
            begin
                Meldung( 'Schriftliche Noten unvollständig' );
                Result := false;
                exit;
            end;

            if FeldWertAnzahl( fC0, 'NoteMdl', '999' ) > 0 then
            begin
                Meldung( 'Mündliche Noten unvollständig' );
                Result := false;
                exit;
            end;

            Result := true;

    // Feststellung der Prüfungsteilleistungen (Folie 8 )
            with fC0 do
            begin       // Durchlauf über alle Fächer
                First;
                while not Eof do
                begin
          Edit;
          FieldByName( 'NotePrfGesamt' ).Clear;     // sicherheitshalber erst mal leeren
          if not FieldByName( 'NoteSchr' ).IsNull or not FieldByname( 'NoteMdl' ).IsNull then
          begin // schriftliche oder mündliche Note vorhanden
            if not FieldByName( 'NoteSchr' ).IsNull and not FieldByname( 'NoteMdl' ).IsNull
            then // MN und SN vorhanden
              FieldByName( 'NotePrfGesamt' ).AsFloat := 0.5*( FieldByName( 'NoteSchr'
              ).AsFloat + FieldByname( 'NoteMdl' ).AsFloat )
            else if not FieldByname( 'NoteSchr' ).IsNull then // SN vorhanden
              FieldByName( 'NotePrfGesamt' ).AsFloat := FieldByName( 'NoteSchr' ).AsFloat
            else if not FieldByname( 'NoteMdl' ).IsNull then // MN vorhanden
              FieldByName( 'NotePrfGesamt' ).AsFloat := FieldByName( 'NoteMdl' ).AsFloat;
          end;
          Post;
                    Next;
                end;
            end;

            idf := NoteVorhanden( fC0, 'NotePrfGesamt', '=', 6, -1 );
            Result := idf = '';
            if not Result then
                Meldung( 'Mindestens eine Prüfungsteilleistung ungenügend' )
            else
            begin
                idf := NoteVorhanden( fC0, 'NotePrfGesamt', '>=', 5, -1 );
                Result := NumToken( idf, ';' ) <= 2;
                if not Result then
```

```
                    Meldung( 'Mindestens zwei Prüfungsteilleistungen mangelhaft' )
            end;

    // Feststellen der Abschlussnoten (Folie9)
            if Result then
            begin
                with fC0 do
                begin       // Durchlauf über alle Fächer
                    First;
                    while not Eof do
                    begin
//            if FieldByname( 'NoteAbschl' ).AsInteger = 999 then
            begin
              if FieldByName( 'NotePrfGesamt' ).IsNull then
              begin
                an := FieldByName( 'Note' ).AsFloat;        // keine Prüfungsteilleistung:
                Abschlussnote = Vornote
                nprfg := 'NULL';
              end else
              begin     // Prüfungsteilleistung vorhanden
                nprfg := StringReplace( FieldByName( 'NotePrfGesamt' ).AsString, ',', '', []
                );
                an := 0.5*( FieldByName( 'Note' ).AsFloat + FieldByName( 'NotePrfGesamt'
                ).AsFloat );
                if frac( an ) > 0 then
                begin       // Endnote nicht "eindeutig"
                  if FieldByname( 'Note' ).AsFloat < an then        // Vornote besser als
                  Abschlussnote ==> Abschlussnote wird besser
                    an := trunc( an )
                  else  // Vornote schlechter als Abschlussnote ==> Abschlussnote wird
                  schlechter
                    an := trunc( an + 1 );
                end;
              end;
              Edit;
              FieldByname( 'NoteAbschl' ).AsInteger := trunc( an );
              Post;

              TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKFaecher SET
              NoteAbschluss=%d, NotePrfGesamt=%s WHERE Schueler_ID=%d AND Fach_ID=%d',
                                  [ trunc( an ), nprfg, fS_ID, FieldByName( 'Fach_ID'
                                  ).AsInteger ] ) );
            end;
            Next;
                    end;
                end;

    // Berechnung des Prüfungsergebnisses (Folie 10)
    // Differenzierungsbereich ist nicht relevant
                AusContainerLoeschen( fC0, FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 ) );

                idf := NoteVorhanden( fC0, 'NoteAbschl', '=', 6, -1 );
                if idf <> '' then
                begin       // mindestens eine 6
                    Meldung( 'Mindestens eine Abschlussnote ungenügend' );
                    Result := false;
                end else
                begin
                    idf := NoteVorhanden( fC0, 'NoteAbschl', '=', 5, -1 );
                    if NumToken( idf, ';' ) > 1 then
                    begin   // mindestens zwei 5
                        Meldung( 'Mindestens zwei Abschlussnoten mangelhaft' );
                        Result := false;
                    end else if NumToken( idf, ';' ) = 1 then
                    begin   // eine 5, suche Ausgleich
                        idf := NoteVorhanden( fC0, 'NoteAbschl', '<=', 3, -1 );
                        if idf = '' then
                        begin
```

```
                        Meldung( 'Eine Abschlussnote mangelhaft, kein Ausgleich
                        gefunden' );
                        Result := false;
                    end;
                end;
            end;
        end;

        if not Result then
        begin       // Abiturprüfung nicht bestanden
            Meldung( 'Abiturprüfung nicht bestanden' );
            TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET
            Bestanden=%s WHERE Schueler_ID=%d',
                                                  [ QuotedStr( '-' ),
                                                  fS_ID ] ) );
            TransactionHandler.DoExecute( Format( 'UPDATE SchuelerLernabschnittsdaten
            SET AbschlussArt=%d, Abschluss=NULL WHERE ID=%d', [ 2, fA_ID] ) );
            TransactionHandler.DoExecute( Format( 'UPDATE Schueler SET EntlassArt=%s,
            Entlassjahrgang_ID=%d WHERE ID=%d', [ QuotedStr( '0A' ), FJahrgang_ID,
            fS_ID] ) );
            exit;
        end else
        begin       // Abiturprüfung bestanden
            TransactionHandler.DoExecute( Format( 'UPDATE SchuelerBKAbschluss SET
            Bestanden=%s WHERE Schueler_ID=%d',
                                                  [ QuotedStr( '+' ),
                                                  fS_ID ] ) );
// Nun die Durchschnittsnote aus den Abschlussnoten ermitteln (außer Religion und Sport)
// Religion und Sport aus C0 raus und in C1 rein
//            Uebertragen( fC0, fC1, FeldWertIDs( fC0, 'Fachgruppe', 'RE', 0 ) );
//            Uebertragen( fC0, fC1, FeldWertIDs( fC0, 'Fachgruppe', 'SP', 0 ) );
// C1 leeren
            fC1.EmptyTable;

            DSN := BK_NotenDurchschnitt( fC0, 0, 'NoteAbschl' );
            DurchschnittsNoteSpeichern( DSN, 'Durchschnittsnote', 'DSN_Text' );
            Meldung( Format( '%s (Durchschnittsnote %s)', [ 'Abiturprüfung bestanden',
            FloatToStr( DSN ) ] ) );
        end;
    end;

// Jetzt prüfen, ob Nachweis 2. FS vorhanden
    if ( fAnzSprachenBK < 2 ) and IstInListe( fTeilPruefOrdKrz,
    'APO-BK-03/D-FOS13/FGHR;APO-BK-11/D-FOS13/FGHR;APO-BK-15/D-FOS13/FGHR' ) then
    begin
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerLernabschnittsdaten SET
        AbschlussArt=%d, Abschluss=%s WHERE ID=%d', [ 1, QuotedStr( fTeilPruefOrdKrz ),
        fA_ID] ) );
//Pfotenhauer: D05 und D06: 0Q wird ersetzt durch 8Q
        TransactionHandler.DoExecute( Format( 'UPDATE Schueler SET EntlassArt=%s,
        Entlassjahrgang_ID=%d WHERE ID=%d', [ QuotedStr( '8Q' ), FJahrgang_ID, fS_ID] ) );
        Meldung( 'Kein Nachweis 2. Fremdsprache vorhanden, fachgebundene Hochschulreife
        erreicht' );
        Result := false;        // Damit Alg. nicht nochmals durchlaufen wird
    end else if ( fAnzSprachenBK >= 2 ) and IstInListe( fTeilPruefOrdKrz,
    'APO-BK-03/D-FOS13/AHR;APO-BK-11/D-FOS13/AHR;APO-BK-15/D-FOS13/AHR' ) then
    begin       // Nachweis vorhanden
        TransactionHandler.DoExecute( Format( 'UPDATE SchuelerLernabschnittsdaten SET
        AbschlussArt=%d, Abschluss=%s WHERE ID=%d', [ 1, QuotedStr( fTeilPruefOrdKrz ),
        fA_ID] ) );
        TransactionHandler.DoExecute( Format( 'UPDATE Schueler SET EntlassArt=%s,
        Entlassjahrgang_ID=%d WHERE ID=%d', [ QuotedStr( '8K' ), FJahrgang_ID, fS_ID] ) );
        Meldung( 'Nachweis 2. Fremdsprache vorhanden, allgemeine Hochschulreife
        erreicht' );
    end;
    Meldung( '' );
end;
```

```
begin
    msg := '';

// Differenzierungsbereich raus
//  AusContainerLoeschen( fC0, FeldWertIDs( fC0, 'Fachgruppe_ID', '30', 0 ) );

    case fBK_Modus[1] of
    '?' : Meldung( 'Die Prüfungsordnung des Schülers/der Schülerin umfasst eine
    Abschlussprüfung. Bitte verwenden Sie die Kartei-Seite "BK-Abschluss".' );
    'Z' : // Zulassung prüfen
                if Zugelassen_D05_D06 then
                    Result := fBK_Modus
                else
                    Result := 'N';
    'M' :   // Mündliche Noten festlegen
                begin
                    ResetBKMdlPruefungen;
                    MuendlichePruefungen_D05_D06;
                end;
    'A' : // Abschluss
                if AbschlussBerechnen_D05_D06 then
                    Result := 'A';
    end;

end;

function TBKPruefungsAlgorithmus.NachpruefungsFaecherErmitteln: boolean;
var
    ds: TDataset;
    i : integer;
    actID : integer;
    fach, res : string;
    cnt: integer;
    notetmp, notesav: double;
    testen: boolean;
    lstIDs: TIntegerList;
    Prf10, cmd: string;
begin
// Die IDs sichern
    lstIDs := TIntegerList.Create;
    try

    fCS.First;
    while not fCS.EOF do
    begin
        lstIDs.Add( fCS.FieldByName( 'ID' ).AsInteger );
        fCS.Next;
    end;

    fNachprFaecherErmitteln := true;
    fMeldungAktiv := false;
    cnt := 0;

    for i := 0 to lstIDs.Count - 1 do
    begin       // Schleife über die Fächer-IDs
// Jedesmal wieder aus fCS nach fC0 zurück
        fC0.EmptyTable;
        fC1.EmptyTable;
        fC2.EmptyTable;
        fCS.First;
        while not fCS.EOF do
        begin
            fC0.Append;
            RBKCopyRecord( fCS, fC0 );
            fC0.Post;
            fCS.Next;
        end;
```

```
        with fC0 do
        begin
            actID := lstIDs[i];
            Locate( 'ID', actID, [] );
            fach := FieldByname( 'FachName' ).AsString;
            prf10 := FieldByname( 'Prf10Fach' ).AsString;
            if fPruefungsArt = 'V' then
            begin   // Bei Versetzung
                testen := FieldByname( 'Note' ).AsFloat = 5;//fGrenznote + 1;
                notetmp := 4;
                notesav := FieldByname( 'Note' ).AsFloat;
            end else if fPruefungsArt = 'A' then
            begin // Bei Abschluss
                testen := FieldByname( 'Note' ).AsFloat > FGrenzNote;
                notetmp := FieldByname( 'Note' ).AsFloat - 1;
                notesav := FieldByname( 'Note' ).AsFloat;
            end;
            if testen and ( prf10 = '-' ) then
            begin
                Edit;
                FieldByname( 'Note' ).AsFloat := notetmp;
                Post;

//------------------------------------------------------------------------------------------------
------
// Die Versetzungen
//------------------------------------------------------------------------------------------------
------
                if IstInListe( fTeilPruefOrdKrz,
                'APO-BK-03/B/V11;APO-BK-11/B/V11;APO-BK-15/B/V11' ) then
                    res := Pruefe_BK_B_Versetzung
                else if IstInListe( fTeilPruefOrdKrz,
                'APO-BK-03/C-BFS-BAB/V;APO-BK-11/C-BFS-BAB/V;APO-BK-15/C-BFS-BAB/V' ) then
                    res := Pruefe_BK_C_Versetzung;

                Locate( 'ID', actID, [] );      // wieder zurückfinden
                if ( ( res = 'V' ) or ( res = 'A' ) ) and ( prf10 = '-' ) then// mit einer 4
                würde die Versetzung/Abschluss klappen
                begin
                    inc( cnt );
                    fMeldungAktiv := true;
                    if cnt = 1 then
                        Meldung( ' ' );
                    Meldung( fach + ' ist mögliches Nachprüfungsfach' );
                    fMeldungAktiv := false;
          if FNPFaecher <> '' then
            FNPFaecher := FNPFaecher + ', ';
          FNPFaecher := FNPFaecher + fach;
                end;
                Edit;
                FieldByname( 'Note' ).AsFloat := notesav;
                Post;
            end;
            Next;
        end;
    end;
    fMeldungAktiv := true;
    fNachprFaecherErmitteln := false;
    Result := cnt > 0;
    if cnt = 0 then
    begin
        Meldung ( ' ' );
    Meldung( 'Keine möglichen Nachprüfungsfächer gefunden' );
    cmd := Format( 'update SchuelerLernabschnittsdaten set MoeglNPFaecher=NULL where ID=%d',
    [ FA_ID ] );
    end else
    cmd := Format( 'update SchuelerLernabschnittsdaten set MoeglNPFaecher=%s where ID=%d', [
    QuotedStr( FNPFaecher ), fA_ID ] );
```

```
  TransactionHandler.DoExecute( cmd );

    finally
      FreeAndNil( lstIDs );
    end;

end;

function TBKPruefungsAlgorithmus.BK_NotenDurchschnitt( Tabelle: TDataset; const doppelt:
integer; const nfld: string ): double;
var
    cnt: integer;
    note: integer;
    gew: integer;
  ares: double;
begin

    Result := 0;
    cnt := 0;
    with Tabelle do
    begin
        First;
        while not EOF do
        begin
//          ShowMessage( FieldByName( 'FachKrz' ).AsString + '=' + FieldByName( 'Fach'
).AsString + ': ' + FieldByname( nfld ).AsString + ', GW: ' +  FieldByName( 'Gewichtung'
).AsString );
            note := FieldByname( nfld ).AsInteger;
            gew := FieldByName( 'Gewichtung' ).AsInteger;
            if ( note > 0 ) and ( gew > 0 ) then
            begin   // Damit "Attest" nicht berücksichtigt wird
                if ( doppelt > 0 ) and ( FieldByname( 'Wochenstd' ).AsInteger >= doppelt )
                then
                begin
                    Result := Result + 2 * note * gew;
                    inc( cnt, 2 * gew );
                end else
                begin
                    Result := Result + note * gew;
                    inc( cnt, gew );
                end;
            end;
            Next;
        end;
    end;
    if cnt > 0 then
    begin
    if DebugHook <> 0 then
    begin // Kontrolle
      ares := Result / cnt;
      ares := trunc( ares*10 );
      ares := ares * 0.1;
    end;

        Result := Result / cnt;
    Result := Result * 10;
        Result := trunc( Result );
        Result := Result * 0.1;

    if DebugHook <> 0 then
      ShowMessage( Format( 'Result=%4.2f, ares=%4.2f', [ Result, ares ] ) );

    end else
        Result := 0;

end;

function TBKPruefungsAlgorithmus.BK_NotenDurchschnittFHR( Tabelle: TDataset; const nfld:
```

```
string ): double;
var
    cnt: integer;
    note: integer;
    gew: integer;
  ares: double;
begin
    Result := 0;
    cnt := 0;
    with Tabelle do
    begin
        First;
        while not EOF do
        begin
            note := FieldByname( nfld ).AsInteger;
            gew := FieldByName( 'GewichtungFHR' ).AsInteger;
            if ( note > 0 ) and ( gew > 0 ) then
            begin   // Damit "Attest" nicht berücksichtigt wird
        Result := Result + note * gew;
        inc( cnt, gew );
            end;
// Neu Mai 2017: Sonderfall für E-Gliederungen: Hier geht auch die schriftliche
FHR-Prüfungsnote in den DS ein
// aber nur, wenn Gewichtung > 0, dann aber immer Gewichtung=1
//Formulierug von Herr Pfotenhauer
//1) Es wurde ein separates Fach mit schriftl. FHR angelegt:
//Dann gehen wir davon aus, dass die Vornote leer ist.
//Dann gehen alle Fächer mit allg.Bild. Gewichtung 1 in den Durchschnitt ein.
//
//2) Es wurde im schriftl FHR Fach kein separates Fach verwendet.
//Dann gehen alle Fächer mit allg.Bild. Gewichtung 1 in den Durchschnitt ein.
//Zusätzlich noch die Vornote aus dem schriftl- FHR Fach!

// Hier braucht nur Fall 2) behandelt zu werden
      if IstEGliederung and ( gew > 0 ) and ( FieldByname( 'NoteSchr' ).AsInteger > 0 ) and
      ( FieldByname( 'Note' ).AsInteger > 0 ) then
      begin  // schrift. Note vorhanden und Vornote>>also Prüfung und Fach in einem
        note := FieldByname( 'Note' ).AsInteger;
        Result := Result + note;
        inc( cnt, 1 );
      end;
            Next;
        end;
    end;
    if cnt > 0 then
    begin
    if DebugHook <> 0 then
    begin // Kontrolle
      ares := Result / cnt;
      ares := trunc( ares*10 );
      ares := ares * 0.1;
    end;

        Result := Result / cnt;
    Result := Result * 10;
        Result := trunc( Result );
        Result := Result * 0.1;

    if DebugHook <> 0 then
      ShowMessage( Format( 'Result=%4.2f, ares=%4.2f', [ Result, ares ] ) );
    end else
        Result := 0;
end;

function TBKPruefungsAlgorithmus.Pruefe_FP( ds: TDataset; const fld: string; const note:
double ): string;
// gGibt die ID's aus, deren Note größer (schlechter) ist als "note"
begin
```

```
    ds.First;
    Result := '';
    while not ds.EOF do
    begin
        if ds.FieldByName( 'Fach' ).AsString = 'FP' then
        begin
            if ds.FieldByname( fld ).AsFloat > note then
            begin
                if Result <> '' then
                    Result := Result + ';';
                Result := Result + ds.FieldByName( 'ID' ).AsString;
            end;
        end;
        ds.Next;
    end;
end;

function TBKPruefungsAlgorithmus.NoteVorhandenGewichtung( Tabelle: TDataset; Feld, Op:
string; Note: double; const Gewichtsfeld: string ): string;
var
    fnd: boolean;
    actnote: double;
  pruefen: boolean;
begin
    Result := '';       // d.h. kein fach mit der Note gefunden
    with Tabelle do
    begin
        First;
        while not EOF do
        begin
      if GewichtsFeld = '' then
        pruefen := true
      else
        pruefen := FieldByname( Gewichtsfeld ).AsFloat > 0;
            if pruefen then
            begin
                actnote := FieldByname( Feld ).AsFloat;
                if op = '=' then
                    fnd := actnote = Note
                else if op = '>' then
                    fnd := actnote > Note
                else if op = '>=' then
                    fnd := actnote >= Note
                else if op = '<=' then
                    fnd := ( actnote > 0 ) and ( actnote <= Note )
                else if op = '<' then
                    fnd := ( actnote > 0 ) and ( actnote < Note );
                if fnd then
                begin
                    if Result <> '' then
                        Result := Result + ';';
                    Result := Result + FieldByName( 'ID' ).AsString;
                end;
            end;
            Next;
        end;
    end;
end;

end.
```

Hinweise zu Gliederung C

C 1: Technische Assistentin/Technischer Assistent und Fachhochschulreife
FHR- und BA-Prüfung

```
C01

C 2: Technische Assistentin/Technischer Assistent für Hochschulzugangsberechtigte
Nur BA-Prüfung (FHR ist Vorauss.)
C02

C 3: Kaufmännische Assistentin/Kaufmännischer Assistent und Fachhochschulreife
FHR- und BA-Prüfung
C01

C 4: Kaufmännische Assistentin/Kaufmännischer Assistent für Hochschulzugangsberechtigte
Nur BA-Prüfung (FHR ist Vorauss.)
C02

C 5: zweijährige Berufsfachschule, erweiterte berufliche Kenntnisse und Fachhochschulreife
FHR- und erw. BK-Prüfung
C03

C 6: einjährigen Lehrgang der Berufsfachschule für Hochschulzugangsberechtigte,
erweiterte berufliche Kenntnisse
Nur erw. BK-Prüfung
C04

C 7: Gymnastiklehrerin/Gymnastiklehrer und Fachhochschulreife
FHR- und BA-Prüfung
C01

C 9: FOS 11 und 12,berufliche Kenntnisse und Fachhochschulreife
FHR-Prüfung
C06

C 10
FHR-Prüfung
C08


C 11
FHR-Prüfung
C07


Folgerung
C01: FHR + BA
C02: nur BA
C03: FHR- und erw. BK-Prüfung
C04: Nur erw. BK-Prüfung
C06: FHR
C07: FHR
C08: FHR
```