Spezifikation für einen guten Programmierstil in Pascal

Es sei klargestellt, dass dieses nur einen gut lesbaren Programmierstil beschreibt. Es wird nicht bestritten, dass es auch andere von vergleichbarer Qualität gibt.

1. Nicht mehr als 80 Zeichen pro Zeile schreiben.

2. Nicht mehr als 1 Befehl pro Zeile. So etwas wie

```
FOR i := 1 TO 10 DO INC(x); test := FALSE;
```

   immer schreiben als

```
FOR i := 1 TO 10 DO
  INC(x);
test := FALSE;
```

3. Leerzeilen nach Zusammengehörenden Abschnitten einfügen zur besseren Übersicht. Bsp.:

```
(* Sortiere Daten *)
REPEAT
  ...
UNTIL NOT getauscht;

(* Suche im Feld *)
REPEAT
  ...
UNTIL NOT gefunden;
```

   Unbedingt mindestens eine Leerzeile zwischen Prozeduren / Funktionen / Objektdeklarationen, ...

4. Das END zu einem BEGIN immer in der gleichen Spalte schreiben. Bei längeren Strukturen entweder mehr in Prozeduren / Funktionen / Objekte auslagern, oder das END mit einem Kommentar versehen, auf welchen BEGIN es sich bezieht. Bsp.:

```
IF FALSE THEN
BEGIN
  DEC(x);
END; (* IF *)
```

5. Einrücken bei Schleifen oder Alternativanweisungen (mindestens 2 Zeichen, aber auch nicht mehr als 5 und vor allem auch nicht in der Anzahl variieren). Bsp.:

```
CASE c OF
'1' :
  BEGIN
    ...
  END;
ELSE
  BEGIN
    ...
  END;
END;
```

6. Kommentar
   a. Standarddokumentation des Programmtextes, z.B. anhand der Textblöcke, die in der Datei "BLOECKE.PAS" zu finden sind (siehe Anhang). Auf jeden Fall gehört ein kurzer, prägnanter Text zu:

- jeder Unit
- jedem Hauptprogramm
- jedem Objekt (zu jeder Klasse)
- jeder Prozedur/Funktion
- jedem unüblichen/trickreichen Mittel, z.B. Programmfragmente in Assembler, ...

Sollte der erklärende Text zu lang werden, so schreibt man ihn in eine Datei mit gleichem Namen mit der Endung .TXT. Dateien mit erklärenden Grafiken u.a. können auch andere Extensions tragen. In die Programmdatei gehört dann ein Verweis auf die erklärenden Dateien. In BP 7.0 gibt es übrigens die Tastenkombination CTRL-ENTER, mit der die Datei geöffnet wird, auf deren Name gerade der Cursor im Texteditor steht, z.B. BLOECKE.PAS (na drück schon die Taste).

b. Mitprotokollieren von Änderungen mit Datum, Autor (wenn mehrere/am besten mit Kürzel). Bei plötzlich auftretenden Fehlern können dann zuerst die Änderungen auf Korrektheit geprüft werden.

c. Nicht nur bei trickreichen Sachen Kommentare einfügen, sondern auch mal ab und zu, so dass ein Unbeteiligter schnell die Stelle im Programmtext findet, die er sucht.

d. Nur eine Kommentarart verwenden, d.h. entweder den Kommentar in (* *) oder in { }. Dann kann man Programmstellen mit Kommentar mit den Kommentarklammern der jeweils anderen Sorte zum Testen auskommentieren. Bsp.:

```
{
  WHILE TRUE DO
  BEGIN
    (* Dies ist ein Test *)
    INC(x);
  END;
}
```

7. Weitere bewährte Verbesserungen des Stils:

a. Nach den Schlüsselwörtern USES, CONST, TYPE, VAR immer erst in der darauffolgenden Zeile weiterschreiben. Bsp.:

```
TYPE
      Adresse = RECORD
        name,
        vorname : STRING [30];
      END;
CONST
      s : STRING = 'Hallo Welt';
```

Außerdem bietet es sich an, Pascal-Schlüsselworte immer groß zu schreiben. Seit es Syntaxcoloring gibt, ist dies nicht mehr unbedingt nötig.

b. Genügend Leerzeichen innerhalb der Zeilen. Bsp.:

```
Alt : s[1]:=z[f(a)-f(b)]+17;

Neu : s [1] := z [f(a)-f(b)] + 17;
```

c. Oft ist es schöner, wenn := oder : in aufeinanderfolgenden Zeilen übereinander stehen. Bsp.:

```
VAR
    index,j : INTEGER;
    zaehler : BYTE;
BEGIN
  index   := j;
  zaehler := index*3;
```

```
END;
```

d. Den Pointer auf eine Struktur immer in der Zeile vorher deklarieren. Arbeitet man mit dynamischen Strukturen, so bietet es sich an, den Pointertyp immer zu deklarieren. Dies kostet keinen Speicher! Bsp.:

```
TYPE
     PFeld = ^TFeld;
     TFeld = ARRAY [1..100] OF INTEGER;
```

e. Vereinheitlichung von Typ-, Variablen-, und Konstantennamen. Die ersten Zeichen des Namens könnten folgende Bedeutung haben:

```
T  : OBJECT-, RECORD- oder ARRAY-Typ
P  : Pointer auf einen T-Typ
V  : Statische Variable eines T-Typs
VP : Pointervariable auf einen T-Typ
```

Bsp.: (TFeld und PFeld wie oben)

```
PROCEDURE test;
VAR
    VPFeld    : PFeld;
    VFeldSwap : TFeld;
BEGIN
  NEW(VPFeld);
  ...
  DISPOSE(VPFeld);
END;
```

Konstantennamen können sich bei der Verwendung mehrerer Units überschneiden. Um sicher zu gehen, dass die richtige benutzt wird, kann man sich die ersten Zeichen der Konstanten reservieren. Davon machen auch Standard-Pascal-Units gebrauch:

```
Windowsunit :
  faReadOnly  : Fileattribut Read-only

Dosunit :
  ReadOnly
```

Des weiteren gilt : lange selbsterklärende Variablennamen haben noch keinem Programm geschadet.

f. Variablen in Deklarationen sollten vom Typ her immer in der gleichen Reihenfolge stehen. Es bietet sich an, sie nach der Größe des belegten Speicherplatzes zu sortieren (bietet gute Übersicht über den belegten Platz auf dem Stack). Da dies nicht immer eindeutig ist, schlage ich folgende Reihenfolge der Grundtypen vor:

```
FILE, TEXT, OBJECT, RECORD, ARRAY, STRING, POINTER, EXTENDED, DOUBLE,
COMP, SINGLE, REAL, LONGINT, WORD, INTEGER, BYTE, SHORTINT, BOOLEAN.
```

Anhang : Mögliches Aussehen von Standardkommentarblöcken

Datei : BLOECKE.TXT

Logische Abschnitte

```
(*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %
 % Abschnitt A :
 %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%*)
```

Prozedurenkopf

```
(*****************************************************************************
 * Name          :
 * Beschreibung  :
 * Input         :
 * Output        :
 * Seiteneffekte :
 *****************************************************************************)
```

Unitkopf / Programmkopf

```
(*---------------------------------------------------------------------------
  | Datei        :
  | Verfasser    : Martin Schneider
  | Datum        :
  | Beschreibung :
  |-------------------------------------------------------------------------
  | Erweiterungen
  |
  | Autor Datum Beschreibung
  |
  |-------------------------------------------------------------------------*)
```

Lokaler Prozedurenkopf

```
  (*/////////////////////////////////////////////////////////////////////////
    / Beschreibung :
    /////////////////////////////////////////////////////////////////////////*)
```

Trennung von lokalen Prozeduren / Funktionen

```
(**********)
```

Objektbeschreibung (eigentlich : Klassenbeschreibung)

```
     (*_____________________________________________________________________
       _
       _
       _____________________________________________________________________*)
```

Die Datei mit diesen Blöcken kann beim Arbeiten mit Borland-Pascal immer im einem Fenster des Editors im Hintergrund liegen. Bei Bedarf markiert man einen Bereich und kopiert ihn mit CTRL-Insert in die Zwischenablage und von dort mit SHIFT-Insert in die Pascaldatei.