18 Strukturierter Datentyp »file« (Dateien)

Gliederung

18.1 Vorbemerkungen, Dateitypen

Unter Datei (engl. file) versteht man eine Sammlung von Daten, die letztlich auf einen Datenträger (Diskette, Platte) geschrieben (gespeichert) werden oder von diesem Datenträger gelesen werden.

Die Datei hat auf dem Datenträger einen Namen nach Konvention des Betriebssystems; übliche Extensionen sind z.B. ».DAT« oder ».TXT«, letztere vorzugsweise für Textdateien.

Programmintern erhält die Datei einen frei wählbaren Dateibezeichner (Dateivariable), nach Pascal-Konvention; häufig wählt man nur kurz »F«. Vor dem ersten Öffnen der Datei ist mit der Prozedur »Assign« (Details siehe später) eine Zuordnung von DOS-Dateiname zu programminternen Dateibezeichner notwendig.

Vor dem Zugriff auf die Datei ist diese zu "öffnen". Dazu dienen die Prozeduren »Reset« oder »Rewrite«. Wenn die Datei im Programm nicht mehr weiter angesprochen wird, ist sie mit der Prozedur »Close« zu "schließen". Das Schließen ist sehr wichtig, da der Dateizugriff "gepuffert" erfolgt. Wird ein Programm bei einer noch geöffneten Datei beendet oder abgebrochen, was z.B. auch durch einen Laufzeitfehler oder durch einen Stromausfall geschehen kann, dann ist ein Datenverlust die Folge.

Innerhalb eines Programms kann die gleiche Datei beliebig oft geöffnet und geschlossen werden.

Es können auch mehrere Dateien gleichzeitig geöffnet sein. Die Anzahl der gleichzeitig geöffneten Dateien wird durch das Betriebssystem bestimmt und beträgt in der Standardeinstellung 8. Für eine größere Anzahl ist der Parameter »files« in der MS-DOS-Konfigurationsdatei CONFIG.SYS auf den gewünschten Wert zu setzen, z.B. mit »*files = 15*«, siehe Kap. 29. Der Höchstwert: »*files = 20*«.

Für das Lesen eines Datensatzes dienen die Prozeduren »Read« oder »ReadLn«, zum Schreiben entsprechend »Write oder »WriteLn«. Bei diesen Prozeduren ist gegenüber den bisher behandelten Anwendungen ein weiterer Parameter – der programminterne Dateibezeichner - anzugeben und zwar als erster Parameter. Besonderheit bei "untypisierten" Dateien siehe später.

Die Dateitypen in Turbo-Pascal

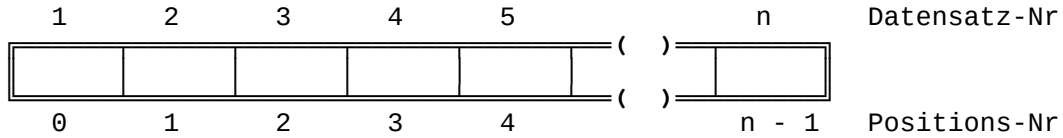
Turbo-Pascal kennt drei Dateitypen:

1) Typisierte Datei mit Datensätzen fester Länge als Komponenten.
2) Textdatei: Datei mit Textzeilen variabler Länge als Komponenten. Sonderfall: Externe Geräte (Bildschirm, Tastatur, Drucker, serielle Schnittstelle) als Textdatei. Dieser Dateityp ist eigentlich auch typisiert, aber anders zu handhaben als 1).
3) Untypisierte Datei. Sonderanwendungen.

Dateityp		Deklaration:
1)	**Typisierte Datei mit Datensätzen fester Länge als Komponenten**	**var** **bezeichner: file of** komponententyp **Beispiel 1: F1: file of** Integer; **2: F2: file of** Real; **3: F3: file of record** ....; ....; **end;**
2)	**Textdatei mit Text–zeilen variabler Länge als Komponenten**	**var** **dateibezeichner: Text;** **{** *In Delphi-Pascal "TextFile" statt "Text"* }
3)	**Untypisierte Datei. Keine Annahme über Typ u. Organisation der Daten. Nur für Sonderanwendungen.**	**var** **dateibezeichner: file;**

1) Bei typisierten Dateien sind alle Datensätze gleich lang; somit kann auf jeden Datensatz wahlfrei direkt zugegriffen werden, und zwar sowohl schreibend oder auch lesend. Typisierte Dateien werden mit »Write« geschrieben und mit »Read« gelesen. Der Zugriff erfolgt über die Datensatznummer (Positionszeiger); die interne Zählung beginnt mit 0. Mit der Prozedur »Seek« kann die gewünschte Position eingestellt werden. Bei jedem Schreib- oder Lesevorgang wird der Positionszeiger automatisch um eins erhöht. Numerische Daten werden in typisierten Dateien binär gespeichert. Andere Bezeichnungen für »typisierte Datei« mit festen Datensatzlängen: **Direktzugriffsdatei**, **Random-Access-Datei**.

Schematische Darstellung einer typisierten Datei:



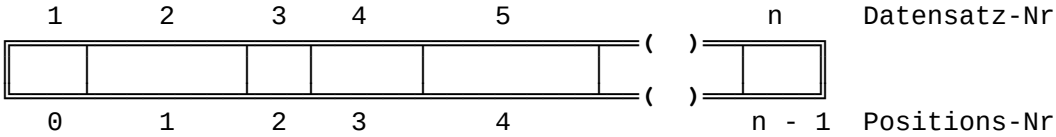
Direkter Zugriff (lesend/schreibend beliebig) auf jeden beliebigen Datensatz.

2) Bei Textdateien sind dagegen die Datensätze unterschiedlich lang. Um auf einen bestimmten Datensatz zugreifen zu können, müssen alle Datensätze vom Dateianfang an durchgegangen werden (sequentieller Zugriff). Quelltextdateien von Programmen sind z.B. immer Textdateien.

Im Gegensatz zu typisierten Dateien kann bei Textdateien die Positionsnummer nicht gewählt werden; dagegen wird der Positionzeiger bei jedem Schreib- und Lesevorgang wie bei typisierten Dateien um eins erhöht. Eine (geöffnete) Textdatei kann entweder nur beschrieben (mit »WriteLn«) oder nur gelesen werden (mit »ReadLn«).

Beim Schreiben wird bei jedem Datensatz der Textdatei an das Ende des Datensatzes automatisch ein *CR* (Carriage Return) als Zeilentrenner geschrieben (genauer: *CR* + *LF*, Carriage Return und Line Feed, #13 und #10); beim Lesen wird bis zu diesen Trennzeichen gelesen.

Schematische Darstellung einer Textdatei:



Nur sequentieller Zugriff. Nur lesend oder schreibend.

Nicht dargestellt sind die Steuerzeichen CR und LF (#13 und #10) am Ende jedes Datensatzes. Manche Texteditoren (so auch der Turbo-Pascal-Editor) erzeugen am Ende der Textdatei noch ein weiteres Steuerzeichen (Zeichen #26, Strg+Z, Ctrl+Z), das auch als Kennung für Dateiende (End of File) interpretiert wird.

3) Bei untypisierten Dateien werden keine Annahmen über die Art und Organisation der Daten gemacht. Damit kann man Dateien mit beliebigen Strukturen bearbeiten, allerdings immer nur ganze Blöcke schreiben (Prozedur »BlockWrite«) oder lesen (Prozedur »BlockRead«). Die Blockgröße beträgt standardmäßig 128 Byte, kann aber zwischen 1 Byte und 64 KByte gewählt werden. Weitere Details siehe Unterpunkt 18.6.

Vorab die wichtigsten Dateiprozeduren und Dateifunktionen

Vollständige Darstellung Kap. 18.3

In den Beispielen wird als Bezeichner für die Dateivariable schlicht »*F*« verwendet. Mit »*a*« sei ein beliebiger Ausdruck und mit »*v*« eine Variable bezeichnet.

- Prozedur »Assign« — Zuordnung der Dateivariablen zum DOS-Dateinamen.
  Beispiel: `Assign(F, 'B:Huber.TXT');`
  Der Pfad ist ein String-Ausdruck
- Prozedur »Reset« — Eine existierende Datei öffnen.
  Beispiel: `Reset(F);`
- Prozedur »Rewrite« — Eine neue Datei anlegen.
  Beispiel: `Rewrite(F);`
- Prozedur »Close« — Datei schließen.
  Beispiel: `Close(F);`
- Prozedur »Write« — In typisierte Datei schreiben
  Beispiel: `Write(F, v);  { v = Variable }`
- Prozedur »WriteLn« — In Textdatei schreiben.
  Beispiel: `WriteLn(F, a); { a = Ausdruck }`

- Prozedur »Read« — Liest einen Wert aus einer typisierten Datei und weist ihn auf eine Variable zu.
  Beispiel: `Read(F, v);`
- Prozedur »ReadLn« — Liest eine Zeile aus einer Textdatei weist sie auf eine Variable zu, in der Regel eine Stringvariable.
  Beispiel: `ReadLn(F, v);`
- Prozedur »Seek« — Setzen des Datei-Positionszeigers bei typisierten Dateien.
  Beispiel: `Seek(F, a);  { a = LongInt-Ausdruck }`
- Funktion »EoF« — End of File. Ergebnistyp ist Boolean
  Beispiel: `EoF(F)`
- Funktion »FileSize« — Liefert die Anzahl der Komponenten (Datensätze) einer typisierter (oder untypisierten) Datei. Nicht für Textdateien. Ergebnistyp ist LongInt.
  Beispiel: `FileSize(F)`

Zur binären Speicherung von numerischen Daten in typisierten Dateien

Es sei eine Integerdatei mit folgenden sechs Integerdaten angenommen:

```
4711  -4711  4712  4711  4712  4711
```

Die Datei als Char-Dump: `g↕Öøh↕g↕h↕g↕`

Der Datei als Hex-Dump: `67 12 99 ED 68 12 67 12 68 12 67 12`

Die einzelnen Speicherdaten in hex und dez, mit Umrechnung in Integerdaten:

```
g↕  67 12   103 018           018 * 256 + 103 =   4711
Öø  99 ED   153 237    (237 - 256)* 256 + 153 =  -4711
h↕  68 12   104 018           018 * 256 + 104 =   4712
g↕  67 12   103 018           018 * 256 + 103 =   4711
h↕  68 12   104 018           018 * 256 + 104 =   4712
g↕  67 12   103 018           018 * 256 + 103 =   4711
```

Es wird zuerst das niederwertige Byte und dann erst das höherwertige Byte abgespeichert. Bei Integerdaten enthält das Bit-Nr 7 (Wertigkeit $2^7 = 128$) das Vorzeichen; die Korrektur ist bei allen höherwertigen Bytes >= 128 in der gezeigten Weise vorzunehmen.

Wird die gleiche Integerdatei (vielleicht versehentlich) als Realdatei eingelesen, dann werden die zwölf Bytes als zwei Reals – ein Real belegt in Turbo-Pascal sechs Byte – interpretiert und zwar wie folgt:

```
 1.7044352770E-08
 1.7043508660E-08
```

18.2 Einleitende Beispiele

18.2.1 Einleitendes Beispiel für typisierte Datei

```
program Pas18021; { Kap. 18.2: Einleitendes Beispiel }
                  { für "Typisierte Datei"           }
uses
  CRT;

const
  DOS_Dateiname = 'C:\Student\Pas18021.DAT';

var
  F:      file of Integer;   { Dateibezeichner, hier kurz "F",     }
                             { ansonsten beliebiger (freier)       }
                             { Bezeichner nach Pascal-Konvention   }
  F_r:    file of Real;
  F_b:    file of Byte;
  F_c:    file of Char;
  F_Bool: file of Boolean;
  i, j:   Integer;
  r:      Real;
  b:      Byte;
  Ch:     Char;
  Bool:   Boolean;

begin
  TextBackground(Blue); ClrScr;

  Assign(F, DOS_Dateiname);   { Mit "Assign" Zuordnung DOS-Dateiname }
                              { auf Dateivariable, hier kurz "F",    }

  TextColor(Yellow);
  WriteLn('Demo "Pas18021.PAS": Typisierte Datei ');
  TextColor(White);

  { ----- 1. Neue Datei anlegen und mit Werten beschreiben --------- }
  Rewrite(F);                  { Mit "Rewrite" neue Datei anlegen.   }
                               { Evtl. vorhandene wird gelöscht.     }
                               { Positionszeiger wird auf Anfang = 0 }
                               { gesetzt. }

  i :=     0;  Write(F, i);    { Nur  V a r i a b l e n  in typi-    }
  j :=  1111;  Write(F, j);    { sierte Datei schreiben. Ausdrücke   }
                               { nur bei Textdateien. In typisierte  }
            { Dateien nur mit "Write" schreiben;  bei Textdateien   }
            { dagegen auch mit "WriteLn". Numerische Werte werden   }
            { typisierten Dateien binär gespeichert, bei Text-      }
            { dateien dagegen als Textzeichen.                      }
  i := 2222;
  j := -4711;
  Write(F, i, j);            { Parameterliste mit mehreren Variablen }
  Close(F);                  { Datei schließen }


  { ----- 2. Datei wieder öffnen und Daten lesen ------------------- }
  Reset(F);                 { Mit "Reset" vorhandene Datei öffnen und }
                            { Positionszeiger auf Anfang = 0 setzen   }
```

```
  Read(F, i);  Write(i, ' ');  { Nur in  V a r i a b l e n einlesen  }
  Read(F, i);  Write(i, ' ');  { (bei allen Dateitypen)             }
  Read(F, i);  Write(i, ' ');  { Bei typisierten Dateien nur "Read", }
  Read(F, i);  Write(i, ' ');  { bei Textdateien nur "ReadLn"        }
  WriteLn; WriteLn;
  Close(F);

  { ----- 3.1 An beliebiger Position lesen ------------------------- }
  Reset(F);
  Seek(F, 2);               { Mit "Seek" Positionszeiger setzen.     }
                            { Hier Position 2 = Datensatz 2 + 1 = 3  }
                            { Positionszeiger: LongInt-Ausdruck      }
  Read(F, i, j);
  WriteLn(i, ' ', j);
  WriteLn;

  { ----- 3.2 An beliebiger Position schreiben/lesen --------------- }
  Seek(F, 3);
  j := 3333;
  Write(F, j);  { Alten Wert überschreiben }

  Seek(F, 3);
  Read(F, i); WriteLn(i);
  WriteLn;

  { ----- 3.3 Alle Daten lesen und Anzahl feststellen -------------- }
  Write('Alle Daten der Datei: ');
  Seek(F, 0);               { Positionszeiger auf Anfang = 0 setzen  }
  j := 0;
  while not EoF(F) do       { Solange Dateiende n i c h t erreicht.. }
    begin                   { Boolean-Funktion EoF = End of File     }
      Read(F, i);
      Write(i, ' ');
      Inc(j);
    end;
  WriteLn;
  WriteLn('Die Anzahl der Daten: ', j);    { ¦ .....: 4 }
  WriteLn('Die Anzahl der Daten mit der Funktion "FileSize(F)": ',
           FileSize(F)); WriteLn;          { ¦ .....: 4 }
  Close(F); { Datei schließen     }

  { ----- 4. Datei beliebig beschreiben und lesen ------------------ }
  Reset(F);                { Mit "Reset" vorhandene Datei öffnen und }
                           { Positionszeiger auf Anfang = 0 setzen   }
  j := 5555;
  i := 5;
  Seek(F, i);
  Write(F, j);
  WriteLn('Auf die Position  ', i, ' wurde der Wert "', j,
          '" geschrieben');

  Seek(F, 4);
  Read(F, j);
  WriteLn('Wert auf Position 4 noch undefiniert, ',
          'hier zufällig: ', j);

  j := 4444;
  Seek(F, 4);
  Write(F, j);
  WriteLn('Auf die Position  4 wurde der Wert "', j, '" geschrieben');
```

```
  WriteLn;

     { Der Lesezugriff auf eine Position hinter der bisherigen }
     { höchsten Schreibposition würde zum Abbruch führen mit   }
     { der Fehlermeldung "Error 100: Disk read error"          }
     { Seek(F, 9);                                             }
     { Read(F, j);                                             }

  Write('Jetzt alle Daten der erweiterten Datei: ');
  Seek(F, 0);
  while not EoF(F) do
    begin
      Read(F, i);
      Write(i, ' ');
    end;
  WriteLn; WriteLn;

  Close(F);

  { ----- 5.1 Integerdatei als Realdatei behandeln ----------------- }
  { Nicht empfehlenswert, hier nur zur Demo. Die Anzahl der Integer- }
  { daten muß für dieses Experiment durch 3 teilbar sein, sonst      }
  { Diskfehler, wenn alle Daten eingelesen werden sollen.            }
  { Zur Erinnerung: Für Integer 2 Byte Speicher, für Real 6 Byte     }

  Assign(F_r, DOS_Dateiname);
  Write('6 Integer auf  2 Real einlesen: ');
  Reset(F_r);
  while not EoF(F_r) do
    begin
      Read(F_r, r);
      Write(r);
    end;
  Close(F_r);
  WriteLn;

  { ----- 5.2 Integerdatei als Bytedatei behandeln ----------------- }
  { Nicht empfehlenswert, hier nur zur Demo. Die 6 Integerdaten      }
  { ergeben 12 Bytedaten.                                            }

  Assign(F_b, DOS_Dateiname);
  Reset(F_b);
  Write('6 Integer auf 12 Byte einlesen: ');
  while not EoF(F_b) do
    begin
      Read(F_b, b);
      Write(b, ' ');
    end;
  WriteLn;
  Close(F_b);

  { ----- 5.3 Integerdatei als Boolean-Datei behandeln ------------- }
  { Nicht empfehlenswert, hier nur zur Demo. Die 6 Integerdaten      }
  { ergeben 12 Boolean                                               }

  Assign(F_Bool, DOS_Dateiname);
  Reset(F_Bool);
```

```
  Write('6 Integer auf 12 Bool einlesen: ');
  while not EoF(F_Bool) do
    begin
      Read(F_Bool, Bool);
      if Bool
         then Write('T ')
         else Write('F ');
    end;
  WriteLn;
  Close(F_Bool);

  { ----- 5.4 Integerdatei als Char-Datei behandeln ---------------- }
  { Nicht empfehlenswert, hier nur zur Demo. Die 6 Integerdaten      }
  { ergeben 12 Char                                                  }

  Assign(F_c, DOS_Dateiname);
  Reset(F_c);
  Write('6 Integer auf 12 Char einlesen: ');
  while not EoF(F_c) do
    begin
      Read(F_c, Ch);
      Write(Ch, ' ');
    end;
  WriteLn;
  { Die Zeichenfolge des Beispiels enthält auch Chr(13);
    damit wird ein CR ausgelöst und der Zeilenanfang überschrieben }
  Close(F_c);

  repeat
  until KeyPressed;
end.
```

```
Demo "Pas18021.PAS": Typisierte Datei

0 1111 2222 -4711

2222 -4711

3333

Alle Daten der Datei: 0 1111 2222 3333
Die Anzahl der Daten: 4
Die Anzahl der Daten mit der Funktion »FileSize(F)«: 4

Auf die Position 5 wurde der Wert "5555" geschrieben
Wert auf Position 4 noch undefiniert, hier zufällig: 29295
Auf die Position 4 wurde der Wert "4444" geschrieben

Jetzt alle Daten der erweiterten Datei: 0 1111 2222 3333 4444 5555

Integer auf Real einlesen:  0.0000000000E+00 5.4990905417E-38
Integer auf Byte einlesen: 0 0 87 4 174 8 5 13 92 17 179 21
Integer auf Bool einlesen: F F T T T T T T T T T T
 ↖ ◄ | § uf Char einlesen:     W ♦ « ♣
```

18.2.2 Einleitendes Beispiel für Textdatei

```
program Pas18022; { Kap. 18.2: Einleitendes Beispiel für "Textdatei" }

uses
  CRT;
```

```
var
  F:  Text;                { Standard-Dateidatentyp "Text" }
  s,                       { Nicht "file of Text"          }
  s1,
  s2: string;
  i:  Integer;

procedure Ueberschrift(s: string);
begin
  TextColor(Yellow);
  WriteLn(s);
  TextColor(White);
end;

begin
  TextColor(Blue); ClrScr;

  Assign(F, 'C:\Student\Pas18022.TXT');
                           { Mit "Assign" Zuordnung DOS-Dateiname    }
                           { auf Dateivariable, hier kurz "F",       }

  { --- Nr 1: ------------------------------------------------------ }
  Ueberschrift('Nr. 1: Neue Textdatei anlegen');
  Rewrite(F);              { "Rewrite": Neue Textdatei zum Schreiben }
                           { anlegen. Wenn Datei bereits vorhanden,  }
                           { dann wird sie überschrieben (gelöscht). }

  s := 'Anton Huber, München ';
  i := 4711;
  WriteLn(F, s);           { Mit "WriteLn" in Textdatei schreiben;   }
                           { "Write" im Prinzip auch möglich, aber   }
                           { auf jeden Fall Abschluß mit "WriteLn".  }
  WriteLn(F, i);           { Bei typisierten Dateien dagegen nur mit }
  WriteLn(F, 'FHM: ' + s); { "Write". In die Textdatei können        }
  WriteLn(F, s, i + 10);   { beliebige Ausdrücke geschrieben werden; }
                           { bei typisierten Dateien dagegen nur     }
                           { Variablen. Nach jeder Zeile einer Text- }
                           { datei wird ein "Return" gespeichert,    }
                           { genauer ein CR + LF, #13 + #10.         }
  Close(F);                { Datei schließen                         }

  { --- Nr. 2: ------------------------------------------------------ }
  Ueberschrift('Nr. 2: Textdatei lesen');
  Reset(F);                { Textdatei zum Lesen öffnen. Im Gegen-   }
                           { satz zu typisierten Dateien können      }
                           { Textdateien entweder nur zum Schreiben  }
                           { oder nur zum Lesen geöffnet werden.     }

  while not Eof(F) do      { Solange nicht am Dateiende ... }
    begin
      ReadLn(F, s);        { Aus Textdatei nur mit "ReadLn" lesen;   }
                           { In der Regel auf Stringvariable. Bei    }
                           { typisierten Dateien dagegen mit "Read"  }

      WriteLn(s);          { Die Bildschirmausgabe:     }
                           { ¦Anton Huber, München      }
                           { ¦4711                      }
                           { ¦FHM: Anton Huber, München }
                           { ¦Anton Huber, München 4721 }
    end;
```

```
  Close(F);                { Datei schließen }

  { --- Nr. 3: ------------------------------------------------- }
  Ueberschrift('Nr. 3: Neue Textdatei anlegen. Numerikdaten');
  Rewrite(F);              { Neue Datei zum Schreiben öffnen.      }
  for i := 11 to 12 do     { Im Gegensatz zu typisierten Dateien   }
    WriteLn(F, i);         { werden numerische Werte in Textdateien }
                           { nicht binär, sondern als Textzeichen  }
                           { gespeichert und können somit beim Aus- }
                           { lesen auch auf String-Variablen zuge- }
                           { wiesen werden.                        }
  Close(F);                { Datei schließen                       }

  { --- Nr. 4: ------------------------------------------------- }
  Ueberschrift('Nr. 4: Textdatei lesen. Numerik auf String');
  Reset(F);                { Datei zu Lesen öffnen }
  while not Eof(F) do
    begin                  { ¦11  }   { Numerische Daten k ö n n e n }
      ReadLn(F, s);        { ¦12  }   { bei Textdateien auch auf     }
      WriteLn(s);          {           Strings eingelesen werden.   }
    end;
  Close(F);                { Datei schließen                       }

  { --- Nr. 5: ------------------------------------------------- }
  Ueberschrift('Nr. 5: Textdatei lesen. Numerik auf Numerik');
  Reset(F);                { Datei zu Lesen öffnen }
  while not EoF(F) do
    begin                  { ¦11  }  { Hier Einlesen von numerischen }
      ReadLn(F, i);        { ¦12  }  { Daten einer Textdatei auf     }
      WriteLn(i);          {          Integervariable.               }
    end;
  Close(F);                { Datei schließen                       }

  { --- Nr. 6: ------------------------------------------------- }
  Ueberschrift('Nr. 6: Textdatei am Ende fortsetzen');
  Append(F);               { (Text-) Datei am Ende fortschreiben }
  WriteLn(F, 'Fortsetzungszeile 1');
  WriteLn(F, 'Fortsetzungszeile 2');
  Close(F);

  { --- Nr. 7: ------------------------------------------------- }
  Ueberschrift('Nr. 7: Erweiterte Textdatei lesen');
  Reset(F);              { Erweiterte Datei komplett lesen }
  while not EoF(F) do
    begin
      ReadLn(F, s);
      WriteLn(s);
    end;
  Close(F);

  { ------------------------------------------------------------- }
  repeat
  until KeyPressed;
end.
```

18.3 Standardprozeduren und -funktionen für Dateien

Die Auflistung verwendet folgende Abkürzungen:

P Prozedur
F Funktion
V Byte-Variable aus der Standard-Unit *SYSTEM*

F Dateibezeichner, Dateivariable
a Ausdruck
v Variable

Die Auflistung enthält auch einige Prozeduren und Funktionen, die zwar nicht direkt mit Dateien zu tun haben, sondern mit dem Datenträger, wie z.B. die Funktion "DiskFree" und somit doch häufig in Verbindung mit Dateien gebraucht werden.

Art	Bezeichner	Bemerkungen, Beispiele
P	Append	Öffnet vorhandene Textdatei für Schreiben weiterer Daten am Dateiende. Beispiel: `Append(F)`
P	Assign	Zuordnung der Dateivariablen zum DOS-Dateinamen. Beispiel: `Assign(F, 'B:Huber.TXT')`
P	AssignCRT	Unit CRT. Zuordnung Bildschirm zu Textdatei für schnellere Bildschirm-Ausgabe als über DOS. Beispiel: `AssignCRT(F)`
P	BlockRead	Liest Record(s) aus untypisierter Datei in Puffervariable. Siehe Demo-Programm.
P	BlockWrite	Schreibt Record(s) aus Puffervariablen in untypisierte Datei. Siehe Demo-Programm.
P	ChDir	Wechselt Standardverzeichnis, ähnlich DOS-Befehl CD Beispiel: `ChDir('A:\Ordner2')`
P	Close	Datei schließen Beispiel: `Close(F)`
F	DiskFree	Unit DOS. Liefert Größe des freien Speichers auf dem angebenen Laufwerk. Ergebnistyp LongInt. Beispiel: `DiskFree(0)` `{ 0 = aktuell. Laufwerk }` Beispiel: `DiskFree(1)` `{ 1 = Laufwerk A }` Beispiel: `DiskFree(2)` `{ 2 = Laufwerk B }` Beispiel: `DiskFree(3)` `{ 3 = Laufwerk C usw. }`
F	DiskSize	Unit DOS. Liefert die Gesamtkapazität des angebenen Laufwerks in Byte. Ergebnistyp LongInt. Beispiel: `DiskSize(1)` `{ 0, 1, 2, 3 wie bei }` `{ Funktion "DiskFree" }`
F	EoF	End of File. Prüfung, ob Dateiende erreicht. Ergebnistyp Boolean.

Beispiel: `EoF(F)`

F EoLn — End of Line. Prüfung, ob Position innerhalb einer Textdatei auf Zeilenende steht. Ergebnistyp Boolean.
Beispiel: `EoLn(F)`

P Erase — Löscht Datei, die aber nicht geöffnet sein darf.
Beispiel: `Erase(F)`

F FExpand — Unit DOS. File Expand. Erweitert den DOS-Dateinamen um den dazugehörigen Pfad.
Beispiel: `FExpand('LiGleich.PAS');`
Das Ergebnis könnte z.B. sein: `C:\STUDENT\LIGLEICH.PAS`

V FileMode — In der Standard-Unit *SYSTEM* deklariert. Legt fest, in welchem Modus typisierte und untypisierte Dateien durch *Reset* geöffnet werden.

```
Beispiel: FileMode := 2; { Standard. Für Lesen }
                         { und Schreiben       }
Beispiel: FileMode := 0; { Öffnen von untypisierten }
                         { Dateien nur zum Lesen    }
Beispiel: FileMode := 1; {Nur für Schreiben }
```

F FilePos — Liefert die momentane Position innerhalb der typisierten oder untypisierten Datei. Zählung ab 0. Ergebnisdatentyp LongInt. Siehe auch Prozedur »Seek«. Für die Byte-Position muß das Ergebnis noch mit Größe der Komponente multipliziert werden, bei Integer z.B. mit dem Faktor 2.
Beispiel: `FilePos(F)`

F FileSize — Liefert die Anzahl der Komponenten (Datensätze) in der typisierten oder untypisierten Datei. Ergebnistyp LongInt. Wenn die Datei leer ist, dann wird das Ergebnis 0 geliefert.
Beispiel: `FileSize(F)`

P FindFirst — Unit DOS. Sucht ein Verzeichnis nach dem ersten Vorkommen eines Dateinamens ab. Details siehe Kap. 18.7 und Online-Hilfe.

P FindNext — Unit DOS. Setzt die mit *FindFirst* begonnene Suche nach einem Dateinamen fort; siehe dort.

P Flush — Erzwingt das Schreiben des Puffers in die Textdatei.
Beispiel: `Flush(F)`

F FSearch — Unit DOS. File Search. Sucht eine Liste von Verzeichnissen nach einem DOS-Dateinamen ab. Details siehe Online-Hilfe.

P	FSplit	Unit DOS. File Split. Zerlegt einen vollständigen DOS-Dateinamen in seine Komponenten: Zugriffspfad, Dateiname und Extension. Details siehe Online-Hilfe.
P	GetDir	Unit DOS. Ermittelt das aktuelle Verzeichnis eines Laufwerks (0 = aktuelles Laufwerk, 1 = A: 2 = B, 3 = C usw.) und legt es unter der Stringvariablen v ab. Beispiel: `GetDir(0, v)  { 0 = aktuelles Laufwerk }`
P	GetFAttr	Unit DOS. Get File Attribut. Liefert die File-Attribute (ReadOnly, Hidden, SysFile usw.) einer Datei unter der Word-Variablen v zurück. Durch anschließende Bit-Operationen können die einzelnen Attribute bestimmt werden. Details siehe Online-Hilfe. Beispiel: `GetFAttr(F, v)`
F	IOResult	Liefert den Fehlerstatus der letzten Ein-/Ausgabeoperation zurück. Ergebnistyp Word. Bei fehlerfreier Ausführung hat IOResult den Wert 0. Siehe Demo-Programm und Kapitel 25. Beispiel: `if IOResult <> 0 then .......`
P	MkDir	Make Direktory. Legt ein neues Unterverzeichnis an, wie DOS-Befehl MD. Beispiel: `MkDir('C:\Student\LiGleich')`
P	Read	Liest Wert(e) aus typisierter Datei in die Variable(n). Beispiel: `Read(F, v)` Beispiel: `Read(F, v1, v2, v3)`
P	ReadLn	Liest Zeile aus einer Textdatei und weist sie auf Variable(n) zu. Beispiel: `ReadLn(F, v)`
P	Rename	Ändern des DOS-Dateinames ähnlich wie DOS-Befehl REN. Der in der Assign-Prozedur genannte alte DOS-Name wird über die Dateivariable `F` durch einen neuen (Stringausdruck) ersetzt, wobei im Gegensatz zu REN auch der Pfad gewechselt werden kann. Die Prozedur darf nicht bei geöffneten Dateien angewendet werden. Beispiel: `Rename(F, DOS_neu);`
P	Reset	Existierende Datei öffnen. Beispiel: `Reset(F)`
P	Rewrite	Neue Datei anlegen. Eine eventuell existierende Datei wird überschrieben. Beispiel: `Rewrite(F)`
P	RmDir	Löscht ein Verzeichnis, das aber leer sein muß. Wirkung wie der entsprechende DOS-Befehl. Beispiel: `RmDir('C:\Ordner1\Ordner11');`

P	Seek	Setzt Positionszeiger einer typisierten oder untypisierten Datei auf die angegebene Komponente n, die einen LongInt-Ausdruck darstellt. Beispiel: `Seek(F, n)`
F	SeekEoF	Prüft bei einer Textdatei, ob sich zwischen der momentanen Position und dem Dateiende noch "echte" Zeichen befinden, wobei Leerzeichen, Tabulatoren und Zeilenvorschub = (CR + LF) nicht betrachtet werden. Ergebnistyp Boolean. Beispiel: `SeekEoF(F)`
F	SeekEoLn	Prüft bei einer Textdatei, ob sich zwischen der momentanen Position und dem nächsten Zeilenvorschub = (CR + LF) noch "echte" Zeichen befinden; siehe auch »SeekEoF«. Ergebnistyp Boolean. Im Gegensatz zur Funktion *EoLn* wird bei *SeekEoLn* der Positionszeiger versetzt. Beispiel: `SeekEoLn(F)`
P	SetFAttr	Unit DOS. Setzt die Attribute der Datei, die mit der Assign-Prozedur mit der Dateivariablen *f* verbunden ist. Das Attribut ist ein Byte-Ausdruck, wobei auch die in der Unit DOS deklarierten Konstanten (ReadOnly, Hidden usw) benutzt werden können. Details siehe Online-Hilfe. Die Datei darf nicht geöffnet sein. Beispiel: `SetFAttr(F, ReadOnly);`
P	SetTextBuf	Mit dieser Prozedur kann die Größe des Puffers einer Textdatei, die standardmäßig 128 Byte beträgt, auf einen beliebige Word-Ausdruck gesetzt werden. Details siehe Online-Hilfe.
P	Truncate	Schneidet eine typisierte oder untypisierte Datei an der momentanen Position ab; d.h. die restlichen Daten werden gelöscht. Beispiel: `Truncate(F);`
P	Write	Wert(e) in typisierte Date schreiben Beispiel: `Write(F, v);` | *v, v1, v2, v3*: Beispiel: `Write(F, v1, v2, v3);` | *Variablen*
P	WriteLn	Zeile in Textdatei schreiben und Abschluß mit Return Beispiel: `WriteLn(F, a)` | *a, a1, a2, a3:* Beispiel: `WriteLn(F, a1, a2, a3)` | *Ausdrücke*

18.4 Weitere Demos für typisierte Dateien

```
program Pas18041; { Kap. 18.4: Demo typisierte Datei }

uses
```

```
  CRT;

const
  DOS_Dateiname = 'C:\Student\Pas18041.DAT';  { Ggf. Pfad ändern     }
  Min           = -4711;            { Später Zufallsdaten aus diesem }
  Max           = -Min;             { Bereich in Datei schreiben.    }
  Anzahl        = 5;                { Für Demo nur wenige Daten.     }

var
  F:       file of Integer;         { Somit für jeden Wert 2 Byte.   }
  Wert:    Integer;
  i:       Word;
  Ch,
  Zugriff: Char;

begin
  TextBackGround(Blue); TextColor(Yellow); ClrScr;

  GotoXY(2, 1); Write('Demonstration: Typisierte Datei');
  TextColor(White);

  Assign(F, DOS_Dateiname);  { Zuweisung des DOS-Dateinamens auf "F" }

{ +-------- A: Neue Datei anlegen, in Datei schreiben -------------+ }
  GotoXY(2, 3);
  TextColor(Yellow); WriteLn('A: Schreiben '); TextColor(White);

  ReWrite(F); { Legt neue Datei an und setzt auch Positionszeiger    }
              { auf Null. Somit "Seek" nicht notwendig, wenn Datei   }
              { - wie hier - von Anfang an geschrieben (oder         }
              { gelesen) wird. }
  for i := 1 to Anzahl do
    begin
      Wert := Min + Random(Max + 1 - Min);  { Für Demo Zufallswerte  }
      Seek(F, i - 1);  { "Seek" in dieser Situation nicht notwendig  }
      Write(F, Wert);  { Mit "Write" (bei typisierten Dateien nicht  }
                       { "WriteLn") Schreiben der  V a r i a b l e n }
                       { "Wert" in die Datei.                        }
      GotoXY(2, WhereY);
      WriteLn(i, ': In Datei:  ', Wert:6);
    end;
  Close(F);

  { +-------- Lesen I (Alle Daten, bei bekannter Anzahl) ----------+ }
  GotoXY(28, 3);
  TextColor(Yellow); WriteLn('B: Lesen I '); TextColor(White);

  Reset(F);  { Öffnet Datei und setzt Positionszeiger auf Null.      }
             { Somit "Seek" nicht notwendig, wenn Datei - wie hier - }
             { von Anfang an gelesen (oder geschrieben) wird.        }
  for i := 1 to Anzahl do
    begin
      Seek(F, i - 1);  { "Seek" in dieser Situation nicht notwendig  }
      Read(F, Wert);   { Einlesen auf  V a r i a b l e  "Wert"       }
                       { Bei typisierten Dateien Einlesen mit        }
                       { "Read", nicht mit "ReadLn".                 }
      GotoXY(28, WhereY);
```

```
    WriteLn('Aus Datei: ', Wert:6);
  end;
Close(F);

{ +--- Lesen II (Alle Daten bis EoF, bei unbekannter Anzahl) ----+ }
GotoXY(51, 3);
TextColor(Yellow); WriteLn('C: Lesen II '); TextColor(White);

Reset(F);  { Öffnet Datei und setzt Positionszeiger auf Null.      }
           { Somit "Seek" nicht notwendig, wenn Datei - wie hier - }
           { von Anfang an gelesen (oder geschrieben) wird.        }

while not Eof(F) do     { "EoF" End of File }
  begin
    Read(F, Wert);
    GotoXY(51, WhereY);
    WriteLn('Aus Datei: ', Wert:6);
  end;
Close(F);

{ +-------- D: Wahlfrei Schreiben und Lesen ---------------------+ }
                        { Nur bei typisierten Dateien möglich }
Reset(F);  { Öffnet Datei und setzt Positionszeiger auf Null. }
           { "Seek" später notwendig, da wahlfrei schreibend  }
           { oder lesend zugegriffen wird.                    }
repeat
  GotoXY(2, 10);
  TextColor(Yellow);
  Write('D: Wahlfreier Zugriff. ',
        '"S" Schreiben, "L" Lesen, "E" Ende: ');
  TextColor(White); ClrEoL;
  repeat
    Zugriff := UpCase(ReadKey);
  until Zugriff in ['S', 'L', 'E'];
  WriteLn(Zugriff);
  if not (Zugriff = 'E') then
     repeat
       GotoXY(2, 11); ClrEoL;
       Write('Datensatz Nr (1..', Anzahl, '): ');
       Ch := ReadKey;
       i  := Ord(Ch) - Ord('0');
     until i in [1..Anzahl];  { Der Positionszeiger zählt   }
                              { aber von 0 bis (Anzahl - 1) }
  case Zugriff of
  'S': begin
         GotoXY(29, 11); Write('Eingabe Integer-Wert: ');
         ReadLn(Wert);
         Seek(F, i - 1);  { Wahlfreier Zugriff, des-   }
         Write(F, Wert);  { halb hier "Seek" notwendig }
       end;
  'L': begin
         Seek(F, i - 1);  { Wahlfreier Zugriff, des-   }
         Read(F, Wert);   { halb hier "Seek" notwendig }
         GotoXY(29, 11); Write('Aus Datei: ', Wert:6);
       end;
  end;
until UpCase(Zugriff) = 'E';
```

```
  Close(F);

  { +-------- E: Lesen III ---------------------------------------+ }
  GotoXY(2, 13);
  TextColor(Yellow); WriteLn('E: Lesen III'); TextColor(White);
  Reset(F);  { Öffnet Datei und setzt Positionszeiger auf Null.      }
             { Somit "Seek" nicht notwendig, wenn Datei - wie hier - }
             { von Anfang an gelesen (oder geschrieben) wird.        }
  i := 0;
  while not Eof(F) do
    begin
      Read(F, Wert);
      Inc(i);
      GotoXY(2, WhereY);
      WriteLn(i, ': Aus Datei: ', Wert:6);
    end;
  GotoXY(2, WhereY + 1);
  WriteLn('Die Anzahl der Datensätze: ', i);
  Close(F);

  repeat
  until KeyPressed;
  { Die Ausgabe unter der Annahme, daß bei Punkt D: der Datensatz
    Nr. 4 mit "S" auf "9999" überschrieben wurde:
    +-----------------------------------------------------------------
    ¦ Demonstration: Typisierte Datei
    ¦
    ¦ A: Schreiben            B: Lesen I           C: Lesen II
    ¦ 1: In Datei:   -4711    Aus Datei:  -4711    Aus Datei:  -4711
    ¦ 2: In Datei:   -2655    Aus Datei:  -2655    Aus Datei:  -2655
    ¦ 3: In Datei:    4603    Aus Datei:   4603    Aus Datei:   4603
    ¦ 4: In Datei:    -858    Aus Datei:   -858    Aus Datei:   -858
    ¦ 5: In Datei:    3752    Aus Datei:   3752    Aus Datei:   3752
    ¦
    ¦ D: Wahlfreier Zugriff. "S" Schreiben, "L" Lesen, "E" Ende:
    ¦ Datensatz Nr (1..5): 4    Eingabe Integer-Wert: 9999
    ¦
    ¦ E: Lesen III
    ¦ 1: Aus Datei:  -4711
    ¦ 2: Aus Datei:  -2655
    ¦ 3: Aus Datei:   4603
    ¦ 4: Aus Datei:   9999
    ¦ 5: Aus Datei:   3752
    ¦
    ¦ Die Anzahl der Datensätze: 5
    ¦
    +-----------------------------------------------------------------
  }
end.
```

```
program Pas18042; { Kap. 18.4:  Demo typisierte Datei }
  {
    Es wird der schreibende und lesende Dateizugriff demonstriert.
    Dabei werden folgende Datei-Prozeduren "P" und -Funktionen "F"
    eingesetzt:

    "Assign"        P   Zuweisung physische Datei an Dateivariable
```

```
    "Rewrite"       P   Neue Datei öffnen und anlegen
    "Close"         P   Datei schließen
    "Reset"         P   Vorhandene Datei öffnen
    "EoF"           F   Ende der Datei. Ergebnistyp: Boolean
    "Seek"          P   Positionszeiger setzen
    "FilePos"       F   Positionszeiger anzeigen
    "FileSize"      F   Anzahl der Datensätze anzeigen
   }
uses
  CRT;
const
  AnzahlDatensaetze = 7;
  Aw = 10; { für spätere Zufallsdaten }
  Ew = 60;
var
  i,
  Zahl,
  Nummer:  Byte;
  Datei:   file of Byte;  { Reserviertes Wort "file", strukturierter }
                          { Typ. Allgemein: "file of Datentyp"       }
                          { Bei Textdateien wegen der variablen      }
                          { Zeilenlängen dagegen: "Datei: Text"      }
begin
  ClrScr;
  Assign(Datei, 'C:\Student\Datei1.DAT');
        { Zuweisung eines physischen (DOS-) Dateinamens, hier        }
        { "C:\Student\Datei1.DAT" an die (logische) Dateivariable,   }
        { hier "Datei". Der physische Dateiname ist ein String-      }
        { Ausdruck nach MS-DOS-Konvention.                           }
  { ---------------------------------------------------------------- }
  Rewrite(Datei);  { Eine neue Datei wird mit "Rewrite" eingerichtet,
                     eine evtl. vorhandene wird zerstört. Vorsicht!
                     Der Positionszeiger wird auf das erste Element
                     gesetzt; die Zählung beginnt aber mit 0.        }
  Write('Test 1:  ');
  for i := 1 to AnzahlDatensaetze do
    begin
      Zahl := Aw + Random(Ew + 1 - Aw);  { Zufallszahl }
      Write(Zahl, ' ');      { ¦Test 1:  10 26 33 26 46 14 36    }
      Write(Datei, Zahl);    { Variable in typ. Datei schreiben. }
                             { Nur Variable, kein Ausdruck!      }
    end;
  WriteLn;
  Close(Datei);  { Schließen mit "Close". Dateien nur so lange offen }
                 { halten wie unbedingt notwendig. Datenverlust bei  }
                 { Stromausfall und bei Rechnerabsturz möglich!      }
  { ---------------------------------------------------------------- }
  Reset(Datei); { Öffnen einer existierenden Datei mit "Reset" }
  Write('Test 2:  ');
  while not EoF(Datei) do      { Standardfunktion "EoF", End of File }
    begin
      Read(Datei, Zahl); { Von Datei lesen und auf Variable zuweisen }
```

```
      Write(Zahl, ' ');  { ¦Test 2:  10 26 33 26 46 14 36 }
    end;
  WriteLn;

  { ---------------------------------------------------------------- }
  Nummer := 3;    { Der 3. "Datensatz" soll überschrieben werden }

  Seek(Datei, Nummer - 1);
     { Der Positionszeiger wird hier mittels "Seek" auf "Nummer - 1" }
     { gesetzt, da die interne Zählung mit 0 beginnt                 }

  Zahl := 99;
  Write(Datei, Zahl);  { Das 3. Element wird überschrieben }

  Seek(Datei, 0);      { Positionszeiger wird auf den Anfang gesetzt }

  Write('Test 3:  ');
  while not Eof(Datei) do
    begin
      Read(Datei, Zahl);
      Write(Zahl, ' ');     { ¦Test 3:  10 26 99 26 46 14 36 }
    end;
  WriteLn;
  { ---------------------------------------------------------------- }
  Nummer := 4;

  Seek(Datei, Nummer - 1);

  Write('Test 4:  ');
  while not EoF(Datei) do
    begin
      Read(Datei, Zahl);
      Write(Zahl, ' ');     { ¦Test 4:  26 46 14 36 }
    end;
  WriteLn;
  { ---------------------------------------------------------------- }
  Nummer := 5;
  Seek(Datei, Nummer - 1);
  Writeln('Test 5a: Der Positionszeiger: ', FilePos(Datei));{ ¦...: 4}

  Write('Test 5b: ');
  Read(Datei, Zahl);
  WriteLn(Zahl);                          { ¦Test 5b: 46  }

  WriteLn('Test 5c: Der Positionszeiger: ', FilePos(Datei));{ ¦...: 5}
  WriteLn('Test 5d: Anzahl Datensätze:   ', FileSize(Datei));{¦...: 7}
  { ---------------------------------------------------------------- }

  Close(Datei);   {  Schließen mit "Close", siehe oben }

  repeat
  until KeyPressed;
end.
```

Das folgende Programm behandelt den Compilerschalter "$I" (automatische IO-Kontrolle und die Standardfunktion IOResult am Beispiel einer typisierten Datei. Die Ausführungen treffen aber für alle Dateitypen zu.

```
program Pas18043; { Kap. 18.4: Demo typisierte Datei  }
  { Weitere Datei-Prozeduren "P" und -Funktionen "F":
    "(Dateiname)"                    P: Löscht eine Datei,    wie DOS
```

```
    "Rename(Dateiname1, Dateiname2)"  P: Umbennen einer Datei, wie DOS
    "IOResult"                        F: Liefert Fehlerstatus der
        letzten Ein-/Ausgabeoperation mit Datentyp Word. Die Funktion
        liefert den Wert 0, wenn kein Fehler aufgetreten ist, sonst
        einen Fehlercode.
        Beispiel: Wenn das Laufwerk nicht betriebsbereit ist, wird der
                  Fehlercode "152" (Drive not ready) geliefert.
     Um auf den Fehler reagieren zu können, muß die automatische IO-
     Kontrolle mit dem Compilerschalter "$I-" abgeschaltet werden.
     Nach einem Fehler werden folgende Ein- und Ausgabeoperationen
     solange ignoriert, bis die Funktion "IOResult" aufgerufen wird.
  }
uses
  CRT;
var
  IO_Fehler: Word;
  F:         file of Char;

begin
  ClrScr;

  Assign(F, 'A:Datei3.DAT');    { Für die Demo soll im Laufwerk A }
                                { keine Diskette sein             }

  {$I-}     { Compilerschalter: Automatische IO-Kontrolle AUS }

  Reset(F);
  IO_Fehler := IOResult;
  {$I+}     { Compilerschalter: Automatische IO-Kontrolle EIN }
  if IO_Fehler = 152
     then WriteLn('Fehler ', IO_Fehler,
                  ': Laufwerk nicht betriebsbereit ... ');
  { ......
    ......
  }
  repeat
  until KeyPressed;
end.
```

18.5 Sonderfall: Geräte als Textdateien

```
program Pas18051;     { Kap. 18.5: Geräte als Textdateien }
  { Alternativ Ausgabe auf Bildschirm, Drucker oder Datei }
uses
  CRT, PRINTER;

var
  Ch, ChL:   Char;
  Textzeile: string;

begin
  TextBackground(Blue); TextColor(Yellow); ClrScr;
  Write('Ausgabe auf Bildschirm (B), Drucker (D) ',
        'oder File (F): ');
  repeat
    Ch := UpCase(ReadKey);
  until Ch in ['B', 'D', 'F'];
  WriteLn(Ch, #13#10);
```

```
  TextColor(White);  { Aber nicht wirksam, wenn mit "Lst" }
                     { auf Bildschirm geschrieben wird    }
  case Ch of
  'B': Assign(Lst, 'CON');  { "Lst": List Device, Standard-    }
  'D': Assign(Lst, 'LPT1'); { Dateibezeichner aus Unit PRINTER }
  'F': Assign(Lst, 'C:\Student\Temp.TXT');
  end;

  Rewrite(Lst);  { "Datei" öffnen für Schreiben }
  WriteLn(Lst, 'Auch zeichenorientierte Geräte wie Bildschirm und  ');
  WriteLn(Lst, 'Drucker können als Textdateien angesprochen werden.');
  WriteLn(Lst, 'Dazu wird "Lst" aus der Unit "Printer" benötigt.   ');
  WriteLn(Lst, '---------------------------------------------------');
  WriteLn(Lst, '"CON":                Console. Bei Ausgabe Bild-   ');
  WriteLn(Lst, '                      schirm, bei Eingabe Tastatur ');
  WriteLn(Lst, '"LPT1", "LPT2", usw:  Paralleler Drucker.          ');
  WriteLn(Lst, '                      Nur für Ausgaben             ');
  WriteLn(Lst, '"PRN":                Wie "LPT1"                   ');
  WriteLn(Lst, '"COM1", "COM2", usw:  Serielle Schnittstellen. Für ');
  WriteLn(Lst, '                      Ein- und Ausgaben.           ');
  WriteLn(Lst, '"AUX":                Wie "COM1"                   ');
  WriteLn(Lst, '"NUL":                Nullgerät. Ignoriert Ausgaben');
  WriteLn(Lst, '                      und liefert bei Eingaben so- ');
  WriteLn(Lst, '                      fort "End of File" (EoF).    ');
  WriteLn(Lst, '                      Für Programm-Testzwecke      ');
  WriteLn(Lst, '---------------------------------------------------',
              #13#10);
  Close(Lst);   { "Datei" schließen  }

  Reset(Lst);   { Öffnen für "Lesen" }
  if Ch = 'B' then
     begin
       TextColor(Yellow);
       Write('Eingabe Textzeile: ');
       Readln(Lst, Textzeile);  { "Lesen" von der Console = Tastatur }
       TextColor(White);
       WriteLn(Textzeile);
     end;
  Close(Lst);
  if Ch = 'F' then
     begin
       TextColor(Yellow);
       Write('Erzeugte Datei wieder löschen (j/n): j');
       GotoXY(WhereX - 1, WhereY);
       repeat
         ChL := UpCase(ReadKey);
         if ChL = #13
            then ChL := 'J';
       until ChL in ['J', 'N'];
       Write(ChL);
       if ChL = 'J'        { Die Prozedure "Erase" löscht Datei  }
          then Erase(Lst); { wie der DOS-Befehl "del Dateiename" }
     end;
  Write(#13#10#13#10, 'Ende mit Tastendruck ... ');
  repeat
  until ReadKey <> '';
end.
```

18.6 Demo untypisierte Datei

```
program Pas18061; { Kap. 18.6: Untypisierte Dateien }
 { Demonstriert das byteweise Einlesen einer (beliebigen) Datei  }
 { in einen Puffer, der anschließend sofort in eine andere Datei }
 { geschrieben wird.                                              }

 { Die Formate von "Reset" und "Rewrite" bei untypisierten Dateien:
   - Reset(  datei [, recordgroesse])
   - Rewrite(datei [, recordgroesse])
   datei:         Dateivariable für typisierte Datei
   recordgroesse: Recordgröße, Word-Ausdruck. Optional.
                  Wenn nicht angegeben, dann wird die Recordgröße
                  auf den Standardwert 128 gesetzt.

   Die Formate von "BlockRead" und "BlockWrite":
   - BlockRead( datei, puffer, rSoll [, rIst])
   - BlockWrite(datei, puffer, rSoll [, rIst])
   datei:  Dateivariable für typisierte Datei
   puffer: Variable beliebigen Typs,
           normalerweise Typ "Byte" oder "Char"
   rSoll:  Word-Ausdruck. Anzahl der Records,
           die gelesen (BlockRead) bzw. geschrieben
           (BlockWrite) werden sollen.
   rIst:   Word-Variable. Optionaler Rückgabewert. Anzahl
           der Records, die tatsächlich gelesen (BlockRead)
           bzw. geschrieben (BlockWrite) wurden.
           Wenn die Option  n i c h t  benutzt wird, dann
           wird ein Laufzeitfehler erzeugt, wenn versucht
           wird, über das Dateiende hinaus Records zu lesen
           (bei BlockRead) oder nicht alle Records geschrieben
           werden können (bei BlockWrite, wenn z.B. Datenträger
           voll ist.
 }

uses
  CRT;

const           { Pfade anpassen }
  DOS_Quelle   = 'Pas18061.QQQ';
  DOS_Ziel     = 'C:\Student\Pas18061.ZZZ';
  Recordlaenge = 1; { byteweise in/aus Puffer }

var
  F_Quelle,
  F_Ziel:     file; { untypisierte Datei      }
  PufferV:    array[1..512] of Byte;
              { Datentyp normalerweise "Byte" oder "Char".    }
              { Die Variable "PufferV" muß mindestens so groß }
              { groß sein wie "Recordlaenge * RecordzahlSoll" }
              { und kann max. 64 KByte groß sein.             }
  rReadSoll,  { Soviele Records sollen gelesen werden         }
  rReadIst,   { Soviele Records wurden tatsächlich gelesen.   }
  rWriteSoll,
  rWriteIst:  Word;

procedure DateiGroessenAnzeigen;
begin
  WriteLn;  { Standardfunktion "FileSize" liefert Anzahl }
```

```
            { der Komponenten. Wenn Datei leer, dann 0.  }
  WriteLn('   Dateigröße Quelle: ', FileSize(F_Quelle));
  WriteLn('   Dateigröße Ziel:   ', FileSize(F_Ziel));
  WriteLn;
end;

begin
  ClrScr;

  Assign(F_Quelle, DOS_Quelle);  Reset(  F_Quelle, Recordlaenge);
  Assign(F_Ziel,   DOS_Ziel  );  Rewrite(F_Ziel,   Recordlaenge);

  DateigroessenAnzeigen;

  rReadSoll := SizeOf(PufferV); { hier auf maximalen Wert setzen }

  repeat
    BlockRead( F_Quelle, PufferV, rReadSoll,  rReadIst);

    Write('   Gelesene Records: ', rReadIst:4); { Nur Demo }
    rWriteSoll := rReadIst;

    BlockWrite(F_Ziel,   PufferV, rWriteSoll, rWriteIst);
    Writeln('   Geschriebene Records: ', rWriteIst:4); { Nur Demo }

    if rWriteIst < rWriteSoll then
       begin
         DateigroessenAnzeigen;
         Write('   Datenträger voll. Abbruch mit Taste Esc ... ');
         Close(F_Quelle);
         Close(F_Ziel);
         repeat
         until ReadKey = #27;
         Halt(4711); { >>>>>>>>>>>> }
       end;
  until (rReadIst = 0);

  DateigroessenAnzeigen;
  Close(F_Quelle);
  Close(F_Ziel);

  Write('   Ende mit Esc ... ');
  repeat
  until ReadKey = #27;
end.
```

18.7 Ausgewählte Datei-Routinen aus der Unit DOS

Inhaltsverzeichnis und Datei-Attribute unter Turbo-Pascal anzeigen:

```
program Pas18071;  { Ausgewählte Datei-Routinen aus der Unit DOS }
                   { K. Haller, 77040599 }
uses
  CRT, DOS;
{ In der Unit DOS ist der Recordtyp "SearchRec" wie folgt definiert:
               +-----------------------------------------------+
               ¦ type                                          ¦
               ¦    SearchRec = record                         ¦
               ¦                  Fill: array[1..21] of Byte;  ¦
```

```
                ¦                  Attr: Byte;                  ¦
                ¦                  Time: LongInt;               ¦
                ¦                  Size: LongInt;               ¦
                ¦                  Name: string[12];            ¦
                ¦                end;                           ¦
                +-----------------------------------------------+
  Zu "Fill":    Für DOS reserviert.
  Zu "Attr":    Dateiattribut, in der Unit DOS wie folgt deklariert:
                +--------------------+
                ¦ const              ¦ dez
                ¦    ReadOnly  = $01; ¦   1
                ¦    Hidden    = $02; ¦   2
                ¦    SysFile   = $04; ¦   4
                ¦    VolumeID  = $08; ¦   8
                ¦    Directory = $10; ¦  16
                ¦    Archive   = $20; ¦  32
                ¦    AnyFile   = $3F; ¦  63 = 1 + 2 + 4 + 8 + 16 + 32
                +--------------------+
                Bei der Eingabe der Attribute ist zu beachten, daß
                die Prozedur "FindFirst" (Anwendung später) auch
                Dateien findet, die ein weniger eingeschränktes
                Attribut besitzen.
   Zu "Time":   Gepackte Darstellung von "Time" und "Date" der Datei-
                erstellung bzw. letzten Änderung. Mit der Unit-DOS-
                Prozedur "UnpackTime" (Anwendung später) kann aufge-
                splittet werden.
   Zu "Size":   Dateigröße in Byte.
   Zu "Name":   Dateiname mit Extension, incl. Trennpunkt.
}
procedure Inhaltsverzeichnis; { --------- Hauptprozedur ------------ }
const
  Standardfarbe = Cyan;
  DirFarbe      = Green;

type
  Str10 = string[10];
  Str8  = string[8];
  Str6  = string[6];
  Str3  = string[3];

var
  Dateiname:     Str8;
  Extension:     Str3;
  DateiInfo:     SearchRec;  { Recordtyp "SearchRec" aus Unit DOS }
  DateiDatum:    Str10;
  DateiZeit:     Str8;
  Attribut:      Byte;
  AttributStr:   Str6;
  i:             Word;
  iStr:          string[3];
  Dir_oder_Vol:  Boolean;
  Kurzform:      Boolean;
  ZeileMax:      Byte;

procedure WriteXY(Spalte, Zeile: Byte; Meldung: string);
begin
  GotoXY(Spalte, Zeile);
  Write(Meldung);
end;
```

```
procedure WarteAufTastendruck(ZeileMax: Byte);
var
  Ch: Char;
begin
  TextColor(DirFarbe);
  WriteXY(25, ZeileMax, ' Weiter mit Tastendruck ... ');
  TextColor(Standardfarbe);
  while KeyPressed do
    Ch := ReadKey;
  Ch := ReadKey;
  ClrScr;
end;

procedure DatumZeit_umformen(DateiInfo:      SearchRec;
                             var DateiDatum: Str10;
                             var DateiZeit:  Str8);
var
  DatumZeit:   DateTime;   {   Recordtpy "DateTime" aus Unit DOS:    }
  JahrStr:     string[4];  {+--------------------------------------+}
  MonatStr:    string[2];  {¦ type DateTime = record                ¦}
  TagStr:      string[2];  {¦                    Year:  1980..2099; ¦}
  StundenStr,              {¦                    Month: 1..12;      ¦}
  MinutenStr,              {¦                    Day:   1..31;      ¦}
  SekundenStr: string[2];  {¦                    Hour:  0..23;      ¦}
                           {¦                    Min:   0..59;      ¦}
begin                      {¦                    Sec:   0..59;      ¦}
  ;                        {¦                  end;                 ¦}
  ;                        {+--------------------------------------+}
  UnpackTime(DateiInfo.Time, DatumZeit);
  { Prozedur "UnpackTime" aus Unit DOS }
  Str(DatumZeit.Year,  JahrStr);
  Str(DatumZeit.Month, MonatStr);
  Str(DatumZeit.Day,   TagStr);
  if Length(TagStr)   = 1 then TagStr   := '0' + TagStr;
  if Length(MonatStr) = 1 then MonatStr := '0' + MonatStr;
  DateiDatum := TagStr + '.' + MonatStr + '.'  + JahrStr;

  Str(DatumZeit.Hour,  StundenStr);
  Str(DatumZeit.Min,   MinutenStr);
  Str(DatumZeit.Sec,   SekundenStr);
  if Length(StundenStr)  = 1 then StundenStr  := '0' + StundenStr;
  if Length(MinutenStr)  = 1 then MinutenStr  := '0' + MinutenStr;
  if Length(SekundenStr) = 1 then SekundenStr := '0' + SekundenStr;
  DateiZeit := StundenStr + ':' + MinutenStr + ':' + SekundenStr;
end;

procedure Dateiname_umformen(DateiInfo:     SearchRec;
                             var Dateiname: Str8;
                             var Extension: Str3);
var
  pPos: Byte;
  Ch:   Char;
  j:    Byte;
begin
  if (DateiInfo.Name = '.') or (DateiInfo.Name = '..')
     then begin
            Dateiname := DateiInfo.Name;
```

```
            Extension := '';
          end
     else begin
            pPos := Pos('.', DateiInfo.Name); { Punktposition }
            if pPos = 0
               then begin
                      Dateiname := DateiInfo.Name;
                      Extension := '';
                    end
               else begin
                      Dateiname := Copy(DateiInfo.Name, 1, pPos - 1);
                      Extension := Copy(DateiInfo.Name, pPos + 1, 3);
                    end;
          end;
  while Length(Dateiname) < 8 do
    Dateiname := Dateiname + ' ';  { Mit Blanks auffüllen }
  while Length(Extension) < 3 do
    Extension := Extension + ' ';  { Mit Blanks auffüllen }

  if (Pos('D', AttributStr) = 0) and (Pos('V', AttributStr) = 0) then
     begin
       for j := 1 to Length(Dateiname) do
         begin
           Ch := Dateiname[j];
           if Ch in ['A'..'Z']                          { In Klein-  }
              then Dateiname[j] := Chr(Ord(Ch) + 32); { buchstaben }
         end;              { Bei ASCII:  32 = Ord('a') - Ord('A') }
       ;
       for j := 1 to Length(Extension) do
         begin
           Ch := Extension[j];
           if Ch in ['A'..'Z']                          { In Klein-  }
              then Extension[j] := Chr(Ord(Ch) + 32); { buchstaben }
         end;
     end;
end;

procedure Drucke_Kopfleiste;
begin
  TextColor(Standardfarbe);
  WriteXY(3, 1, '+-----------------------------------' +
                '------------------------------+');
  WriteXY(3, 2, '¦                                   ' +
                '                              ¦');
  WriteXY(3, 3, '+-----------------------------------' +
                '------------------------------¦');
  WriteXY(3, 4, '¦ Nr¦ Name   ¦Ext¦Attribute¦Dateigröße' +
                '¦DateTime ¦ Datum    ¦ Zeit   ¦');
  WriteXY(3, 5, '+---+--------+---+---------+----------' +
                '+---------+----------+--------¦');
  TextColor(DirFarbe);
  WriteXY(5, 2, 'Directory mit Turbo-Pascal-Unit DOS.  ' +
                'Dr. K. Haller, FHM, 77040599');
  TextColor(Standardfarbe);
  WriteLn;
  Window(1, 6, 80, 25);
end;
```

```
procedure Daten_ausdrucken;  { Der Einfachheit halber  }
begin                        { keine Parameterübergabe }
  if Dir_oder_Vol
     then TextColor(DirFarbe)
     else TextColor(Standardfarbe);
  WriteXY( 4, WhereY, iStr);
  WriteXY( 8, WhereY, Dateiname);
  WriteXY(17, WhereY, Extension);
  GotoXY(21,  WhereY); Write(Attribut:2);
  WriteXY(24, WhereY, AttributStr);
  GotoXY(31,  WhereY);
  if not Dir_oder_Vol                    { D- und V-Einträge haben }
     then Write(DateiInfo.Size:10)       { "Dateigröße" 0. Statt   }
     else if Pos('D', AttributStr) <> 0  { dessen Hinweis drucken. }
             then Write('•Sub--Dir' + #17)   { Zeichen #17 nicht    }
             else Write('•VolumeID' + #17);  { mit "Alt-17" möglich }

  GotoXY( 42, WhereY);  Write(DateiInfo.Time);
  WriteXY(52, WhereY, DateiDatum);
  WriteXY(63, WhereY, DateiZeit);
  TextColor(Standardfarbe);
  WriteXY( 3, WhereY, '¦');   WriteXY( 7, WhereY, '¦');
  WriteXY(16, WhereY, '¦');   WriteXY(20, WhereY, '¦');
  WriteXY(30, WhereY, '¦');   WriteXY(41, WhereY, '¦');
  WriteXY(51, WhereY, '¦');   WriteXY(62, WhereY, '¦');
  WriteXY(71, WhereY, '¦');   WriteLn;
end;

procedure AttributeErmitteln(var AttributStr:  Str6;
                             var Dir_oder_Vol: Boolean;
                             DateiInf0:        SearchRec);
begin
  Attribut := DateiInfo.Attr;
  AttributStr := '';
  { Nachfolgend Bit-Nr 0, 1, 2, 3, 4 und 5 testen, ob gesetzt:   }
  if (Attribut and  1) =  1 then AttributStr := AttributStr + 'R';
  if (Attribut and  2) =  2 then AttributStr := AttributStr + 'H';
  if (Attribut and  4) =  4 then AttributStr := AttributStr + 'S';
  if (Attribut and  8) =  8 then AttributStr := AttributStr + 'V';
  if (Attribut and 16) = 16 then AttributStr := AttributStr + 'D';
  if (Attribut and 32) = 32 then AttributStr := AttributStr + 'A';
  while Length(AttributStr) < 6 do       { String vorne mit }
    AttributStr := ' ' + AttributStr;    { Blanks auffüllen }
  if (Pos('V', AttributStr) <> 0) or (Pos('D', AttributStr) <> 0)
     then Dir_oder_Vol := True
     else Dir_oder_Vol := False;
end;

function KurzformEinzug: Boolean;
var
  Ch: Char;
begin
  ClrScr;
  WriteXY(10, 5, 'Directory mit Routinen der Turbo-Pascal-Unit DOS');
  WriteXY(10, 7, '1  Langfassung. Ausgabe aller DateiInformationen');
  WriteXY(10, 8, '2  Kurzfassung. Keine H-, S-, V- und D-Einträge ');
  WriteXY(10, 9, '-----------------------------------------------');
  WriteXY(10,10, '1'); GotoXY(WhereX - 1, WhereY);
```

```
  repeat
    Ch := ReadKey;
    if Ch = #13
       then Ch := '1'
  until Ch in ['1', '2'];
  Write(Ch);
  if Ch = '1'
     then KurzformEinzug := False
     else KurzformEinzug := True;
  ClrScr;
end;

begin  { Beginn Rumpf der Hauptprozedur "Inhaltsverzeichnis" ------- }
  Kurzform := KurzformEinzug;

  if not Kurzform
     then begin
            ClrScr;
            Drucke_Kopfleiste;
            i        :=  0;  { Für Zähler Anzahl der Einträge }
            ZeileMax := 20;  { Nur für Tastendruck-Hinweis    }
          end
     else begin
            ClrScr;
            Textcolor(DirFarbe);
            Write('  ---------------- Directory ohne H-, S-, ' +
                  'V- und Ordner-Einträge -------------');
            TextColor(Standardfarbe);
            ZeileMax := 24;  { Nur für Tastendruck-Hinweis }
          end;
  FindFirst('C:\*.*', AnyFile, DateiInfo);
     {  Prozedur "FindFirst" aus Unit DOS,                          }
     {  Format:   FindFirst(pfad, dateiattribut, recordvariable)  }
  while DosError = 0 do
    begin      { Variable "DosError" aus Unit DOS. Die Werte:     }
      ;        {  0: Fehlerfrei                                    }
      ;        {  2: Directory nicht gefunden                      }
      ;        { 18: Keine weiteren Einträge (nur bei "FindNext") }
      AttributeErmitteln(AttributStr, Dir_oder_Vol, DateiInfo);
    if not Kurzform
         then begin
                Inc(i);
                Str(i, iStr);
                while Length(iStr) < 3 do
                  iStr := '0' + iStr;
                DatumZeit_umformen(DateiInfo, DateiDatum, DateiZeit);
                Dateiname_umformen(DateiInfo, Dateiname, Extension);
                Daten_ausdrucken;
              end
         else if (Pos('H', AttributStr) = 0) and   { nicht Hidden    }
                 (Pos('S', AttributStr) = 0) and   { nicht System    }
                 (Pos('V', AttributStr) = 0) and   { nicht VolumeID  }
                 (Pos('D', AttributStr) = 0)       { nicht Directory }
                 then Write(DateiInfo.Name:16);    { "normale" Datei }
      if WhereY = ZeileMax
         then WarteAufTastendruck(ZeileMax);
      FindNext(DateiInfo);  { Prozedur "FindNext" aus Unit DOS,  }
```

```
                              { Format:   FindNext(recordvariable) }
    end;
  if not Kurzform
     then WriteXY(3, WhereY, '+----- Ende ----------------------'    +
                              '-----------------------------kha---+')
     else WriteLn(#13,  '  ------- Ende ----------------------'   +
                             '---------------------------------kha----');
  WarteAufTastendruck(ZeileMax);
  Window(1, 1, 80, 25);
end;  { ---- von Hauptprozedur "Inhaltsverzeichnis" --------------- }

begin { ==== Hauptprogramm ======================================== }
  Inhaltsverzeichnis;
end.  { ============================================================ }
```

Eine Ausgabe des Programms Pas18071.PAS (Langfassung der Dateieinträge)**:**

Directory mit Turbo-Pascal-Unit DOS. Dr. K. Haller, FHM, 77040599

Nr	Name	Ext	Attribute	Dateigröße	DateTime	Datum	Zeit
001	DISK_1		40 VA	►VolumeID◄	574832814	03.02.1997	08:05:28
002	RECYCLED		22 HSD	►Sub—Dir◄	557345737	24.09.1996	13:30:18
003	WIN95		16 D	►Sub—Dir◄	574900977	04.02.1997	09:23:34
004	FONTS		16 D	►Sub—Dir◄	574901168	04.02.1997	09:29:32
005	bootlog	txt	34 HA	15752	574904371	04.02.1997	11:01:38
006	command	com	32 A	95382	521686597	24.08.1995	09:50:10
007	suhdlog	dat	35 RHA	7798	542016700	14.02.1996	16:37:56
008	config	sys	32 A	178	574898992	04.02.1997	08:25:32
009	msdos	---	34 HA	22	542016520	14.02.1996	16:32:16
010	BEETH5TH		16 D	►Sub—Dir◄	574845812	03.02.1997	14:27:40
011	PLUGPLAY		16 D	►Sub—Dir◄	542082180	15.02.1996	16:36:08
012	PROGRA~1		17 RD	►Sub—Dir◄	542073555	15.02.1996	12:22:38
013	SCSI		16 D	►Sub—Dir◄	542081263	15.02.1996	16:07:30
014	setuplog	txt	34 HA	42872	542016807	14.02.1996	16:41:14
015	XING		16 D	►Sub—Dir◄	542130734	16.02.1996	08:17:28
016	msdos	sys	7 RHS	1653	574899075	04.02.1997	08:28:06
017	bootlog	prv	34 HA	16085	574904000	04.02.1997	10:54:00
018	detlog	txt	38 HSA	70657	542018991	14.02.1996	17:45:30
019	netlog	txt	32 A	489	542016625	14.02.1996	16:35:34

Weiter mit Tastendruck ...