Turbo Pascal deel 5

Erik van Bilzen

MSX Club Magazine 38

Scanned, ocr'ed and converted to PDF by HansO, 2001

In deze aflevering komen bestanden en records aan de orde. Tussen het ontwikkelen van de Game And Music Editor (G.A.M.E.) door heb ik nog net tijd gehad om een aflevering te schrijven.

Bestandsbeheer

Turbo Pascal 3.0 kent verschillende soorten bestanden: getypeerde, ongetypeerde en tekstbestanden. In tegenstelling tot BASIC kent Turbo Pascal alleen sequentiële files en géén random access files (bestanden die met FIELD worden gedefinieerd). Dat wil zeggen dat alle gegevens uit een TP-bestand achtereenvolgens gelezen of geschreven moeten worden. Het is niet mogelijk om direct een gegeven in het midden van een bestand te lezen of te schrijven. De 3 soorten bestanden kunnen als volgt worden gedefinieerd:

```
VAR
  Bestl: FILE OF GegevensType
  Best2: FILE; Best3: Text;
```

Bestl is een getypeerd bestand, Best2 een ongetypeerd bestand en Best3 een tekstbestand. Elk bestand moet worden gekoppeld aan een bestandsnaam op diskette. Dit gebeurt door middel van de Assign-procedure. Vervolgens moet een bestand worden geopend met behulp van Reset of Rewrite. Reset wordt gebruikt om uit een bestand te lezen, Rewrite om naar een bestand te schrijven. Hieronder volgen drie voorbeelden van een BASIC-regel met hun TP-variant

BASIC:
OPEN "bestandl.dat" FOR INPUT AS#1

Turbo Pascal:
Assign (Bestl, 'bestandl.dat'); Reset (Bestl);

BASIC:
OPEN "bestand2.dat" FOR OUTPUT AS#1

Turbo Pascal:
Assign (Best2,'bestand2.dat'); Rewrite (Best2);

BASIC:
OPEN "bestand3.dat" FOR APPEND As#l

Turbo Pascal:
Assign (Best3,'bestand3.dat') ; Append (Best3);

Tekstbestanden

Tekstbestanden bevatten regels tekst die worden afgesloten met een carriage return (ASCII-code 13) en een line feed (ASCII-code 10). Er kan uit een tekstbestand worden gelezen met Read en ReadLn (in BASIC resp. INPUT #1 en LINE-INPUT #1) als het bestand is geopend met Reset. Er kan naar een tekstbestand worden geschreven met Write en WriteLn (in BASIC PRINT #l) als het bestand is geopend met Rewrite of Append.

VOORBEELD TEKSTBESTAND

```
VAR
   Best: Text;
   Tekst: STRING[30];

BEGIN
   Assign (Best,'test.dat'); ReWrite (Best);
   Tekst:='Voorbeeld van tekstbestand';
   WriteLn (Best,Tekst);
   Reset (Best);
   ReadLn (Best,Tekst);
   WriteLn (Tekst);
   Close (Best)
END.
```

Met Close wordt het bestand gesloten. Het is niet mogelijk om rechtstreeks een constante naar een bestand te schrijven. De volgende regel geeft dus een foutmelding:
WriteLn (Best,'Voorbeeld van tekstbestand'),-
Om alle regels uit een tekstbestand te lezen kan gebruik worden gemaakt van de Eof-functie (End Of File). Het resultaat van deze functie is van het type BOOL-EAN en heeft de waarde TRUE als het einde van het bestand is bereikt. Zie het voorbeeld bovenaan deze pagina.

Ongetypeerde bestanden

Voor ongetypeerde bestanden en getypeerde bestanden geldt in grote mate hetzelfde als voor tekstbestanden. Er zijn enkele belangrijke verschillen.Voor ongetypeerde bestanden maakt het niet uit of ze worden geopend
met Reset of Rewrite. Er wordt gelezen en geschreven met behulp van de procedures Block-Read en BlockWrite. Deze procedures lezen of schrijven een aantal blokken van 128 bytes groot van of naar een buffer. Ze kunnen bijvoorbeeld worden gebruikt om schermpagina's in te lezen. Na het volgende programma bevat de variabele-array Buffer de eerste 128 bytes van het bestand TEST.DAT.

VAR

```
  Best: FILE; Buffer:ARRAY [0..127] OF BYTE;

BEGIN
  Assign (Best,'test.dat'); Reset (Best);
  BlockRead (Best,Buffer,1);
 Close (Best)
END.
```

De derde parameter van de BlockRead-procedure (1) bevat het aantal blokken dat gelezen moet worden. Er kan verder nog een optionele parameter worden doorgegeven die het resultaat, het werkelijk aantal gelezen blokken, aangeeft. Bijvoorbeeld, een bestand is 10 blokken groot en er wordt opdracht gegeven 20 blokken te lezen.

BlockRead (Best, Buffer, 20, Resultaat);

De variabele Resultaat bevat na afloop de waarde 10. Na het lezen van 10 records is de End-Of-File status bereikt.

VOORBEELD EOF

```
VAR
   Best: Text;
   Tekst: STRING[80];

BEGIN
   Assign (Best,'test.dat'); Reset(Best);
   WHILE not Eof(Best) DO
      BEGIN
         ReadLn (Best,Tekst);
         WriteLn (Tekst)
      END;
   Close (Best)
END.
```

Getypeerde bestanden

Getypeerde bestanden zijn bestanden van een bepaald gegevenstype, bijvoorbeeld:

```
VAR
  Bestl:FILE OF INTEGER;
  Best2:FILE OF REAL;
  Best3:FILE OF BOOLEAN;
```

Een bestand kan ook van een zelf gedefinieerd type zijn, bijvoorbeeld een array of enumeratietype:

```
TYPE
  Kleuren = (rood,oranje, groen);
```

```
VAR
  Best:   FILE  OF  Kleuren; Verkeerslicht:Kleuren;
```

Er kan alleen worden gelezen of geschreven met behulp van Read en Write (niet met ReadLn en WriteLn). Het volgende voorbeeld gaat verder op het voorgaande:

```
BEGIN
  Assign   (Best,'kleuren.dat') ; ReWrite   (Best);
  Verkeerslicht:=rood; Write   (Best,Verkeerslicht) ;
  Close   (Best)
END.
```

Bestanden zullen echter vaker gebruikmaken van een ander, nog niet besproken, gegevenstype, namelijk het RECORD-type.

Records

Naast de tot nu toe behandelde gegevenstypen (integers, reals, bytes, booleans, enumeratietypen en arrays) bestaan er ook record-typen. Een record is een gegevenstype dat bestaat uit een combinatie van verschillende velden die van een willekeurig gegevenstype kunnen zijn. Bij een klantenbestand kan zo'n record bijvoorbeeld bestaan uit een klantnummer, naam, adres, woonplaats en besteed bedrag.

Een voorbeeld van zo'n recorddefinitie

VOORBEELD RECORDDEFINITIE

```
TYPE
   KlantRec = RECORD
                 KlantNr:  INTEGER;
                 Naam    :  STRING[20];
                 Adres   :  STRING[20];
                 Woonpl  :  STRING[20];
                 Bedrag  :  REAL
              END;
```

Een recorddefinitie wordt afgesloten met een END;. Vervolgens kunnen variabelen van het type KlantRecord worden gedefinieerd, zoals in voorbeeld l onderaan deze pagina.

VOORBEELD VARIANTRECORD

```
TYPE
   Landcode = (Binnenland,Buitenland);
   KlantRec = RECORD
                 KlantNr: INTEGER;
                 Naam, Adres, Woonpl: STRING[20];
                 CASE Land:Landcode OF
                    Binnenland: (Bedrag: REAL);
                    Buitenland: (Bedrag: REAL;
                                 Landnaam: STRING[20];
                                   Valuta: STRING[5]);
              END;

VAR
   Land: Landcode;
   Klant; KlantRec;
```

Aan de variabele Klant kunnen gegevens worden toegekend zoals weergegeven in voorbeeld 2. Hetzelfde geldt voor de variabele KlantArray, zoals in voorbeeld 3 is te zien. Tot slot kan een Klant naar een bestand worden geschreven zoals in voorbeeld 4. Het toewijzen van waarden aan een recordvariabele kan op een snellere manier met het WTTH ... DO statement, zoals voorbeeld 5 laat zien.

VOORBEELD EOF

```
VAR
   Best: Text;
   Tekst: STRING[80];

BEGIN
   Assign (Best,'test.dat'); Reset(Best);
   WHILE not Eof(Best) DO
      BEGIN
         ReadLn (Best,Tekst);
         WriteLn (Tekst)
      END;
   Close (Best)
END.
```

Turbo Pascal kent ook de mogelijkheid om de vorm van een record af te laten hangen van een bepaald gegeven. Dat kan met behulp van een CASE-statement in de recorddefinitie. Komt in ons voorbeeld bijvoorbeeld de klant uit het buitenland, dan willen we ook graag de naam van het land en de valuta weten. Een voorbeeld van zo'n variantrecord is weergegeven op de volgende pagina. Door in het programma de waarde van de variabele Land te veranderen (bijvoorbeeld: Land:-Buitenland),

wordt de vorm van het record veranderd.

VOORBEELD 1

```
VAR
   Klant: KlantRec;
   KlantArray: ARRAY [0..10] OF KlantRec;
   KlantBest: FILE OF KlantRec;
```

VOORBEELD 2

```
   Klant.KlantNr:=12;
   Klant.Naam:='De Biteur';
   Klant.Adres:='Credietlaan 7';
   Klant.Woonpl:='Nieuwkoop';
   Klant.Bedrag:=1234.56;
```

VOORBEELD 3

```
   KlantArray[1].KlantNr:=12;
   KlantArray[2]:=Klant
```

VOORBEELD 4

```
   Assign(KlantBest,'klanten.dat');
   Rewrite (KlantBest);
   Write(KlantBest,Klant);
   Close (KlantBest)
```

VOORBEELD 5

```
   WITH Klant DO
      BEGIN
         Klantnr:=12;
         Naam:='De Biteur';
         Enz....
      END;
```

Procedures en functies

We zetten de besproken en nieuwe standaardprocedures en -functies op 'n rij. Optionele parameters staan tussen vierkante haken.

PROCEDURE Append (VAR Beet);
Opent een bestand voor toevoegen. Als een bestand nog niet bestaat, moet het eerst worden gemaakt met Rewrite;

PROCEDURE Assign (VAR Best,Bestandsnaam: STRING);
Kent de naam van een bestand op diskette toe aan een bestandsvariabele.

PROCEDURE Erase (VAR Best);
Verwijdert een ongeopend bestand.

PROCEDURE Rename (VAR Best; NieuweNaam: STRING);
Verandert de naam van een ongeopend bestand in NieuweNaam.

PROCEDURE BlockRead (VAR BestFILE; VAR Buffer; Index: INTEGER [;VAR Resultaat INTEGER]);
Zet Index of minder blokken uitt ongetypeerd Best in Buffer. De optionele parameter Resultaat bevat het werkelijk aantal gelezen blokken.

PROCEDURE BlockWrite (VAR BestFILE; VAR Buffer;
Index: INTEGER [;VAR Resultaat INTEGER]);
Schrijft Index of minder blokken uit Buffer naar ongetypeerd Best. De optionele parameter Resultaat bevat het aantal blokken dat feitelijk naar het bestand is weggeschreven.

PROCEDURE Close (VAR Best);
Sluit een bestand.

FUNCTION Eoln (VAR BestText): BOOLEAN;
Geeft de waarde TRUE als de bestandspointer zich aan het einde van een regel of bestand bevindt. Alleen voor tekstbestanden.

FUNCTION SeekEoh (VAR BestText): BOOLEAN;
Als Eoln, maar negeert alle spaties en tabs.

FUNCTION Eof (VAR Best):BOOLEAN;
Geeft de waarde TRUE als de bestandspointer zich aan het einde van het bestand bevindt of als het bestand leeg is.

FUNCTION SeekEof (VAR Best): BOOLEAN;
Als Eof, maar negeert alle spaties, tabs en Eoln's.

FUNCTION FilePos (VAR Best): INTEGER;
Geeft als resultaat het recordnum-mer, bloknummer of elementnummer van een geopend bestand. Bij ongetypeerde bestanden het bloknummer, bij getypeerde recordbestanden het recordnum-mer en bij overige getypeerde bestanden het

elementnummer. FilePos kan niet worden toegepast op tekstbestanden.

FUNCTION FileSize (VAR Best): INTEGER;
Geeft als resultaat het aantal records, blokken of elementen van een geopend bestand, uitgezonderd tekstbestanden.

PROCEDURE Flush (VAR BestText);
Schrijft de inhoud van de interne bestandsbuffer direct naar een tekstbestand.

PROCEDURE Reset (VAR Best);
Opent het bestand Best voor invoer. Opent ongetypeerde bestanden voor in- en uitvoer.

PROCEDURE Rewrite (VAR Best);
Opent het bestand Best voor uitvoer. Opent ongetypeerde bestanden voor in- en uitvoer. Als het bestand reeds bestaat wordt het verwijderd en opnieuw gemaakt! Gebruik dus Append om gegevens toe te voegen of maak gebruik van een tweede bestand om gegevens te veranderen (l invoer-bestand en l uitvoerbestand met de veranderde gegevens).

PROCEDURE Read[Ln] (VAR Best; v1 [,v2,..., vn]);
Leest één of meer variabelen uit een bestand. ReadLn kan alleen worden toegepast op tekstbestanden, Read op getypeerde- en tekstbestanden.

PROCEDURE Write[Ln] (VAR Best; v1 [,v2,.... vn]);
Schrijft één of meer variabelen naar een bestand. WriteLn kan alleen worden toegepast op tekstbestanden, Write op getypeerde-én tekstbestanden.

PROCEDURE Seek (VAR Best; Nummer. INTEGER);
Verplaatst de bestandspointer van het geopende bestand Best naar het met Nummer aangegeven record-, blok- of elementnummer. Kan niet worden toegepast op tekstbestanden. De opdracht Seek (Best,FüeSize(Best)) zet de pointer één record voorbij het einde van een bestand, zodat vanaf hier gegevens kunnen worden toegevoegd.

PROCEDURE Truncate (VAR Best);
Verwijderd alle records, blokken of elementen voorbij de bestandspointer.