8. Bestanden

8.1 Het type File

Als je alle voorgaande hoofdstukken hebt bestudeerd, dan weet je eigenlijk al heel veel van programmeren met Turbo Pascal. Eén ding is alleen een beetje vervelend. Als je gegevens ingevoerd hebt, zijn die weer verdwenen op het moment dat je de computer uitzet. Want zodra de computer uitgezet wordt, is alles wat in het geheugen stond verdwenen. Tot nu toe hebben we alleen maar in het geheugen van de computer gewerkt. In dit hoofdstuk leer je hoe je gegevens op schijf kunt bewaren om ze later weer in het geheugen te kunnen inlezen. Ook leer je wat de betekenis van het type File is.

Het type File is bestemd om een verbinding te leggen tussen je computer en de buitenwereld. Hiertussen loopt een gegevensstroom. De buitenwereld kan een bestand op je harde schijf, op een diskette of op een magneetband zijn. Het kan ook een gegevensstroom naar je printer zijn, of gegevens die ingevoerd worden via het toetsenbord. Alle gegevensstromen naar bestanden en aangesloten apparaten worden verzorgd door files die in een programma gewoon als variabelen worden gedeclareerd.

Turbo Pascal kent drie soorten files:

- getypeerde files;
- ongetypeerde files;
- tekstfiles.

Bij een getypeerde file geef je bij de declaratie op met wat voor soort gegevens de file moet werken. De volgende file is bestemd om een stroom integers te behandelen:

```
VAR
  IntegerFile: File OF Integer;
```

Als we het record Ledenkaart uit het vorige hoofdstuk naar een bestand zouden willen schrijven, hebben we niet alleen een file nodig om een verbinding te maken tussen het geheugen en een bestand op de schijf, maar de file moet ook nog weten hoeveel ruimte er in een bestand nodig is om een ledenkaart te kunnen wegschrijven. De declaratie zou dan kunnen zijn:

```
VAR
  LedenkaartFile: File OF Ledenkaart;
```

LedenkaartFile is nu toegerust om gegevens van het type Ledenkaart te vervoeren. Hiermee is LedenkaartFile een getypeerde file die uitsluitend gegevens van het type Ledenkaart afhandelt.

Bij een ongetypeerde file staat het niet van tevoren vast wat voor soort gegevens door de file moeten worden afgehandeld. De afmeting van de af te handelen gegevens kunnen tijdens de uitvoering van het programma worden vastgesteld. Ongetypeerde files worden vaak gebruikt om de schijf te beheren. Hier kun je het kopiëren, verwijderen of hernoemen van bestanden onder verstaan.

Bij een ongetypeerde file blijft bij de declaratie de aanduiding van de typering achterwege:

```
VAR
  Bestand: File;
```

Hier wordt een variabele Bestand gedeclareerd. Het is van tevoren niet duidelijk hoe groot de gegevensblokken zijn die in het bestand staan. Tijdens de uitvoering van een programma kan de grootte van de uit lezen of weg te schrijven blokken worden bepaald.

Tekstfiles zijn bedoeld om regels tekst in een bestand te zetten. Ze kunnen als volgt gedeclareerd worden:

```
VAR
  TekstBestand: Text;
```

Het type Text is binnen Turbo Pascal als volgt gedefinieerd:

```
TYPE
Text = File OF Char;
```

Een regel tekst wordt altijd afgesloten met letterteken 10 en 13 uit de ASCII-tabel. Deze tekens worden respectievelijk "line feed" en "carriage return" genoemd. Line feed betekent "een regel opschuiven", carriage return stamt nog uit de tijd van de ouderwetse schrijfmachine en betekent "wagen naar het begin van de regel". Een file die als type Text gedeclareerd is, werkt dus met regels tekst.

8.2 Assign

Als we een variabele van het type File gedeclareerd hebben, moet er een verbinding worden gelegd met een bestand op de schijf. De verschillende records van het type Ledenkaart worden dan in dit bestand bewaard. Het is als het ware een soort kaartenbak. Laten we eens aannemen dat we dit bestand LEDEN.DAT willen noemen. De verbinding wordt dan gelegd met:

```
Assign(LedenkaartFile,'LEDEN.DAT');
```

Assign zorgt er dus voor dat LedenkaartFile weet met welk bestand op schijf gecommuniceerd moet worden. Omdat LedenkaartFile een file van het type Ledenkaart is, kan hij records uit het bestand op schijf naar het geheugen van de computer transporteren, en weer terug.

8.3 Rewrite en Reset

Voor we in het bestand kunnen schrijven of daaruit kunnen lezen, moet de file eerst geopend worden. Er zijn twee manieren om een file te openen. De eerste manier is:

Rewrite(LedenkaartFile);

Rewrite schrijft een nieuw, leeg bestand naar schijf. Het enige wat dit bestand bevat, is een End Of File-teken. Als er al een bestand met dezelfde naam op schijf stond, zijn de gegevens in dat bestand reddeloos verloren, omdat het bestand overschreven wordt met het nieuwe, lege bestand. Wees dus erg voorzichtig met het gebruik van Rewrite.

De tweede manier om een bestand te openen is:

Reset(LedenkaartFile);

Reset opent een file voor een reeds bestaand bestand. Dit bestand wordt niet overschreven, maar intact gelaten. Zodoende kan het bestand geïnspecteerd of gemuteerd worden. Het bestand is nu gereed om erin te schrijven.

8.4 FileSize

We willen een record aan een bestand toevoegen. Hiertoe moeten we weten hoe groot dat bestand is. De Turbo Pascal-functie FileSize geeft het aantal eenheden aan dat in het bestand geschreven is. Als we een file van het type Byte hebben, krijgen we het aantal bytes terug dat in het bestand staat. Als we schrijven:

Aantal := FileSize(LedenkaartFile);

krijgen we het aantal records van het type Ledenkaart terug. Dit aantal hebben we nodig om een nieuw record op de juiste plaats in het bestand weg te schrijven.

8.5 EOF

Bij het aanmaken van het bestand wordt door de Turbo Pascal-procedure Rewrite in het nieuwe bestand een End Of File-teken gezet. Als we een bestand record voor record lezen, dan moeten we weten wanneer we het einde van het bestand bereikt hebben. Dit kunnen we bepalen met de Turbo Pascal-functie EOF. Deze functie is van het type Boolean en retourneert een True als het einde van het bestand bereikt is. Met de declaratie:

```
IF EOF(LedenkaartFile) THEN ...
```

krijgt EOF de file als parameter mee. De functie retourneert False zolang het eindee van het bestand niet bereikt is, en retourneert True zodra het einde wel bereikt is.

8.6 De bestandswijzer

Om te bepalen waar in een bestand gelezen of geschreven moet worden, heeft het type File de beschikking over een bestandswijzer. Als een file geopend wordt, dan wijst deze bestandswijzer naar de eerste positie in het bestand. Dit is positie 0. Als we vervolgens een rij records zouden willen lezen, dan kan dit in een lus:

```
Assign(LedenkaartFile,'LEDEN.DAT');
Reset(LedenkaartFile);
WHILE NOT EOF(LedenkaartFile) DO
   Read(LedenkaartFile,Ledenkaart);
```

In deze lus wordt het bestand LEDEN.DAT van het begin tot het einde uitgelezen. Bij iedere leesactie wordt de bestandswijzer automatisch één record verder gezet. Deze leesmethode heet sequentieel lezen; telkens wordt het volgende record gelezen. In de volgende paragrafen zullen we zien dat je zowel sequentieel kunt lezen als schrijven.

8.7 Seek en FilePos

Bij sequentieel lezen of schrijven wordt de bestandswijzer telkens na iedere lees- of schrijfactie verhoogd. We kunnen met behulp van de Turbo Pascal-procedure Seek de bestandswijzer ook zelf positioneren.

We willen een ledenkaart aan het einde van het bestand toevoegen en hebben de grootte van het record in de variabele AANTAL gezet. Met de procedure Seek kunnen we nu de bestandswijzer naar de plek laten wijzen waar het nieuwe record geplaatst moet worden:

Seek(LedenkaartFile,AANTAL)

In de variabele AANTAL stond het aantal records dat teruggeven werd door de functie FileSize. De bestandswijzer wijst nu naar het einde van het bestand.

In plaats van de tussenstap met AANTAL, hadden we ook kunnen schrijven:

Seek(LedenkaartFile,FileSize(LedenkaartFile))

Functies mogen net als variabelen als parameter worden meegegeven. Deze faciliteit voor het werken met bestanden wordt ook wel "random access" genoemd.

Soms kan het nodig zijn dat je weet op welke positie de bestandswijzer staat. Hiervoor kunnen we gebruik maken van de Turbo Pascal-functie FilePos. Deze functie geeft aan op welke positie de bestandswijzer van een bepaalde file staat. De opdracht:

POSITIE := FilePos(LedenkaartFile)

geeft de positie van de bestandswijzer van LedenkaartFile terug.

8.8 Write en Read

Om nu het record daadwerkelijk in het bestand te schrijven zetten we:

Write(LedenkaartFile,LedenKaart);

Hiermee wordt op de door de bestandwijzer aangegeven plaats een record van het type Ledenkaart weggeschreven. Het lezen van records verloopt op dezelfde wijze. Als je weet dat de gegevens van meneer Jansen opgeslagen zijn in het vijfde record, dan zet je de bestandswijzer op vier en geef je vervolgens een leesopdracht. Het vijfde record heeft nummer 4 omdat het tellen bij 0 begint.

Het lezen van een ledenkaart gaat als volgt:

Read(LedenkaartFile,Ledenkaart);

Als we een lees- of schrijfopdracht geven terwijl we al aan het einde van het bestand zijn, dan wordt de uitvoering van het

programma onderbroken met foutmelding 100, wat inhoudt dat er een fout gemaakt is met de schijf. Die melding krijg je ook als je met Seek de bestandswijzer na het einde van het bestand gepositioneerd hebt.

8.9 Close

Er kunnen binnen MS-DOS maar een beperkt aantal files tegelijkertijd geopend zijn. Daarom moeten we files na gebruik weer sluiten. Dit gaat als volgt:

Close(LedenkaartFile)

Deze opdracht sluit de file LedenkaartFile. Als je vergeet de file af te sluiten, loop je de kans dat het programma onderbroken wordt door foutmelding 18: "Too many files open", wat inhoudt dat je teveel bestanden tegelijkertijd geopend hebt.

8.10 Samenvatting stappen

Uit eigen ervaring weet ik hoe lastig het is de diverse stappen te overzien om met bestanden te kunnen werken. Vandaar dat ik de stappen die ik hiervoor beschreven heb nog even in de juiste volgorde herhaal:

- Maak een type dat in een bestand gezet moet worden. Meestal zal dit een record zijn.

- Declareer een variabele van het type File. Typeer deze file als een file van het type Record waarin de gegevens kunnen staan die je wilt lezen of schrijven.

- Verbind met behulp van Assign de file met het bestand op schijf.

- Open de file met Reset of Rewrite. Dit is afhankelijk van de vraag of je een bestaand bestand wilt openen of een nieuw bestand wilt maken.

- Gebruik Seek om de bestandswijzer te positioneren.

- Lees met Read het gezochte record of schrijf met Write het record in het bestand.

- Sluit het bestand met Close.

8.11 Voorbeeld getypeerde file

Dit onderwerp wordt waarschijnlijk het beste duidelijk aan de hand van een programma met toelichting. In het programma FILES_1 gaan we een simpele telefoonklapper maken. De klapper moet namen en telefoonnummers kunnen bewaren, we moeten erin kunnen bladeren en gericht kunnen zoeken. Bovendien moet het mogelijk zijn een record te veranderen. Als je het programma over gaat typen, begin dan bij het hoofdprogramma. Typ daarna de verschillende functies en procedures in de volgorde waarop ze in het hoofdprogramma worden aangeroepen. Op deze manier kun je beter zien hoe het programma loopt:

```
PROGRAM FILES_1;
USES CRT;

TYPE
   TTelefoon = Record
      Naam    : String[30];
      Voornaam: String[15];
      Tussen  : String[10];
      Nummer  : String[20];
   END;

   TTelFile = File Of TTelefoon;

PROCEDURE Boodschap(Zin:String);
VAR
   CH: Char;
BEGIN
   ClrScr;
   GotoXY(3,5);
   Write(Zin);
   CH := ReadKey
END;

FUNCTION Bestaat(FileNaam:String):Boolean;
VAR
   F: File;
BEGIN
   Assign(F,FileNaam);
   {$I-}
   Reset(F);
   Close(F);
   {$I+}
   Bestaat := IOResult = 0
END;

FUNCTION Hoofdletters(Naam:String):String;
VAR
   I: Byte;
BEGIN
   FOR I := 1 TO Length(Naam) DO
   Naam[I] := Upcase(Naam[I]);
   Hoofdletters := Naam
END;

PROCEDURE Telefoonscherm(VAR Telefoon:TTelefoon;
                         Titel:String);
```

```
VAR
   X, Y: Byte;
BEGIN
   ClrScr;
   X := 5;
   Y := 2;
   GotoXY(X,Y);
   Write(Titel);
   Inc(Y,2);
   GotoXY(X,Y);
   WITH Telefoon DO


   BEGIN
      Write('Naam            : ',Naam);
      Inc(Y,2);
      GotoXY(X,Y);
      Write('Tussenvoegsel   : ',Tussen);
      Inc(Y,2);
      GotoXY(X,Y);
      Write('Voornaam        : ',Voornaam);
      Inc(Y,2);
      GotoXY(X,Y);
      Write('Telefoonnummer  : ',Nummer)
   END
END;

PROCEDURE LeesNummer(VAR Telefoon:TTelefoon;Titel:String);
VAR
   X, Y: Byte;
   NAAM_OK, NUMMER_OK: Boolean;
BEGIN
   REPEAT
      Y := 4;
      X := 23;
      Telefoonscherm(Telefoon,Titel);
      WITH Telefoon DO
      BEGIN
         GotoXY(X,Y);
         Readln(Naam);
         Inc(Y,2);
         GotoXY(X,Y);
         Readln(Tussen);
         Inc(Y,2);
         GotoXY(X,Y);
         Readln(Voornaam);
```

```
         Inc(Y,2);
         GotoXY(X,Y);
         Readln(Nummer);
         NAAM_OK   := Naam > '';
         NUMMER_OK := Nummer > ''
      END;
      IF NOT NUMMER_OK AND NOT NAAM_OK THEN
      Boodschap
      ('Vul een naam en een nummer in s.v.p.')
      ELSE
      IF NOT NUMMER_OK THEN
      Boodschap('Er moet een nummer worden ingevuld.')

      ELSE
      IF NOT NAAM_OK THEN
      Boodschap('Er moet een naam worden ingevuld.')
   UNTIL NAAM_OK AND NUMMER_OK
END;

FUNCTION ZoekOp(VAR TelFile:TTelFile;
         VAR Telefoon:TTelefoon;Titel:String):Integer;
VAR
   NAAM: String;
   CH  : Char;
BEGIN
   ClrScr;
   GotoXY(5,8);
   Write('Geef Naam op .... ');
   Readln(Naam);
   NAAM := Copy(Hoofdletters(Naam),1,5);
   CH := #0;
   Reset(TelFile);
   WHILE NOT EOF(TelFile)
   AND NOT (CH IN [#27,'J','j']) DO
   BEGIN
      Read(TelFile,Telefoon);
      IF NAAM = Hoofdletters(Copy(
      Telefoon.Naam,1,Length(Naam))) THEN
      BEGIN
         Telefoonscherm(Telefoon,Titel);
         GotoXY(5,12);
         Write('Is dit het gezochte nummer ?   ');
         CH := ReadKey
      END
   END;
   IF CH IN ['J','j'] THEN ZoekOp := FilePos(TelFile) - 1
```

```
   ELSE
   ZoekOp := -1;
   Close(TelFile)
END;

PROCEDURE VoerNummerIn(VAR TelFile:TTelFile);
VAR
   TELEFOON: TTelefoon;
BEGIN
   Reset(TelFile);
   FillChar(TELEFOON,SizeOf(TELEFOON),0);
   LeesNummer(TELEFOON,
   'I N V O E R E N   T E L E F O O N N U M M E R S');
   Seek(TelFile,FileSize(TelFile));
   Write(TelFile,TELEFOON);
   Close(TelFile)
END;

PROCEDURE Bladeren(VAR TelFile:TTelFile);
VAR
   TELEFOON: TTelefoon;
   X,Y     : Byte;
   CH      : Char;
BEGIN
   Reset(TelFile);
   WHILE NOT EOF(TelFile) AND (CH<>#27) DO
   BEGIN
      Read(TelFile,TELEFOON);
      Telefoonscherm(TELEFOON,
      'B L A D E R E N  I N  N U M M E R S');
      GotoXY(5,12);
      Write('Esc = naar menu');
      CH := ReadKey
   END;
   Close(TelFile)
END;

PROCEDURE Zoeken(VAR TelFile:TTelFile);
VAR
   TELEFOON: TTelefoon;
BEGIN
   IF ZoekOp(TelFile,TELEFOON,
            'Z O E K E N  N U M M E R. ') = -1 THEN
   Boodschap
   ('De gezochte naam zit niet in het bestand.')
END;
```

```
PROCEDURE Mutatie(VAR TelFile:TTelFile);
VAR
   TELEFOON: TTelefoon;
   POSITIE : Integer ;
   NM      : String;
BEGIN
   POSITIE := ZoekOp(TelFile,TELEFOON,
            'M U T A T I E   N U M M E R ');
   IF POSITIE > -1 THEN
   BEGIN
      GotoXY(5,12);
      ClrEol;
      WITH TELEFOON DO
      BEGIN
         NM := '';
         IF Voornaam > ''THEN NM := NM + Voornaam + ' ';
         IF Tussen > '' THEN NM := NM + Tussen + ' ';
         NM := NM + Naam
      END;
      FillChar(TELEFOON,SizeOf(TELEFOON),0);
      LeesNummer(TELEFOON,'M U T A T I E '+ NM);
      Reset(TelFile);
      Seek(TelFile,POSITIE);
      Write(TelFile,TELEFOON);
      Close(TelFile)
   END
   ELSE
   Boodschap
   ('De gezochte naam zit niet in het bestand.')
END;

PROCEDURE Menu;
VAR
   TELFILE: TTelFile;
   KEUZE  : Char;
   X,Y    : Byte;
BEGIN
   Assign(TELFILE,'KLAPPER.DAT');
   IF NOT Bestaat('KLAPPER.DAT') THEN
   BEGIN
       Rewrite(TELFILE);
       Close(TELFILE)
   END;
   KEUZE := #0;
   TextBackground(0);
   ClrScr;
```

```
   Window(15,6,65,19);
   TextBackground(4);
   TextColor(14);
   REPEAT
      ClrScr;
      X := 5;
      Y := 2;
      GotoXY(X,Y);
      Write
      (’M E N U   T E L E F O O N K L A P P E R ’);
      Inc(Y,2);
      GotoXY(X,Y);
      Write(’ 1 = Invoeren telefoonnummers.’);
      Inc(Y,1);
      GotoXY(X,Y);
      Write(’ 2 = Bladeren in klapper.’);
      Inc(Y,1);
      GotoXY(X,Y);
      Write(’ 3 = Zoek telefoonnummer’);
      Inc(Y,1);
      GotoXY(X,Y);
      Write(’ 4 = Mutatie telefoonnummer’);
      Inc(Y,2);
      GotoXY(X,Y);
      Write(’ 9 = Einde Programma.’);
      KEUZE := ReadKey;
      CASE KEUZE OF
         ’1’ : VoerNummerIn(TELFILE);
         ’2’ : Bladeren(TELFILE);
         ’3’ : Zoeken(TELFILE);
         ’4’ : Mutatie(TELFILE)
      END;
   UNTIL KEUZE = ’9’
END;

BEGIN
   MENU
END.
```

Regels:Toelichting:

[1]3-9Definieer het type voor het record TTelefoon.

[2]10 Declareer een file als type van het record voor telefoonnummers.

11-19 Procedure Boodschap(Zin:String).

14-19Maak het venster schoon en schrijf de boodschap, die in de parameter Zin ontvangen is, aan de gebruiker. Wacht tot een toets wordt ingedrukt.

[6]20-30 Function Bestaat(FileNaam:String):Boolean;

21-22 Declareer een ongetypeerde file.

24 Verbind de file met de parameter FileNaam.

25 Schakel de onderbreking bij fouten uit.

26-27 Open en sluit het bestand.

28 Schakel de onderbreking bij fouten weer in.

29 Geef de functie de waarde False als IOResult ongelijk is aan nul.

[16] 31-38 **Function Hoofdletters(Naam:String):String;**

34-38Zet de letters in de elementen van de parameter Naam om naar hoofdletters met behulp van de Turbo Pascal-functie Upcase. Geef de functie Hoofdletters de waarde die in Naam staat.

[11] 39-64 Procedure Telefoonscherm(VAR Telefoon:TTelefoon;
Titel:String)

43-64Veeg het scherm schoon en plaats de tekst in het venster om records in te kunnen voeren.

[10] 65-100 Procedure LeesNummer(VAR Telefoon:TTelefoon;
Titel:String);

66-68 Declareer de benodigde lokale variabelen.

70-99Ga een REPEAT-lus in die duurt tot de lokale variabelen NAAM_OK en NUMMER_OK de waarde True hebben.

[11] 73 Roep de procedure Telefoonscherm aan.

[12] 74-86Lees, met de cursor op de aangegeven plaatsen, de velden van de VAR-parameter Telefoon in.

[13] 90-98Test of er een naam en een nummer ingevoerd is. Ontbreekt er iets, vermeld dit dan via de procedure Boodschap.

[16] 101-131Function ZoekOp(VAR TelFile:TTelFile;
VAR Telefoon:TTelefoon;Titel:String):Integer;

103-105 Declareer de benodigde lokale variabelen.

107-110Veeg het venster schoon en lees de lokale variabele Naam in.

111 Zet de letters in Naam om naar hoofdletters door de procedure Hoofdletters aan te roepen.

112 Geef CH een beginwaarde, ofwel initialiseer CH.

113 Open TelFile.

114-126Ga een WHILE-lus in die duurt tot het einde van het bestand bereikt is, of tot de Escape-toets is ingedrukt, of tot er in de lokale variabele CH een "J" of een "j" staat.

117 Lees een record in Telefoon.
118-125 Controleer of het aantal letters dat in Naam staat overeenkomt met het aantal letters in Telefoon.Naam. Als dat het geval is, vraag dan of dit het gezochte nummer is. Als het antwoord een "N" of een "n" is, of de naam komt niet overeen, ga dan terug in de lus en lees het volgende record.
127-129 Als het nummer gevonden is, krijgt ZoekOp het nummer van de plaats waar het gezochte record zich in het bestand bevindt. Als het record niet gevonden is, of er is op de Escape-toets gedrukt, dan krijgt ZoekOp de waarde -1.
130 Sluit TelFile.

132-143 Procedure VoerNummerIn(VAR TelFile:TTelFile);

[9] 133-134 Declareer de lokale variabele TELEFOON.
136 Open TelFile.
137 Zet de velden in TELEFOON op 0.
[10] 138-139 Roep de procedure LeesNummer aan.
[14] 140 Zet de bestandswijzer aan het einde van het bestand.
141 Schrijf het record in het bestand.
142 Sluit TelFile.

[15] 144-161 Procedure Bladeren(VAR TelFile:TTelFile)

145-148 Declareer de benodigde lokale variabelen.
150 Open TelFile.
151-159 Ga een WHILE-lus in die duurt tot het einde van het bestand bereikt is of tot de Escape-toets is ingedrukt.
153-158 Lees een record uit het bestand en laat het door het aanroepen van de procedure Telefoonscherm in het venster verschijnen. Wacht op het indrukken van een toets. Als de ingedrukte toets niet de Escape-toets is en ook het einde van het bestand nog niet bereikt is, lees dan het volgende record.

[16] 162-170 Procedure Zoeken(VAR TelFile:TTelFile)

163-164 Declareer de lokale variabele TELEFOON.
165-170 Roep de functie ZoekOp aan. Als deze -1 bevat, roep dan de procedure Boodschap aan.

[17] 171-200 Procedure Mutatie(VAR TelFile:TTelFile)

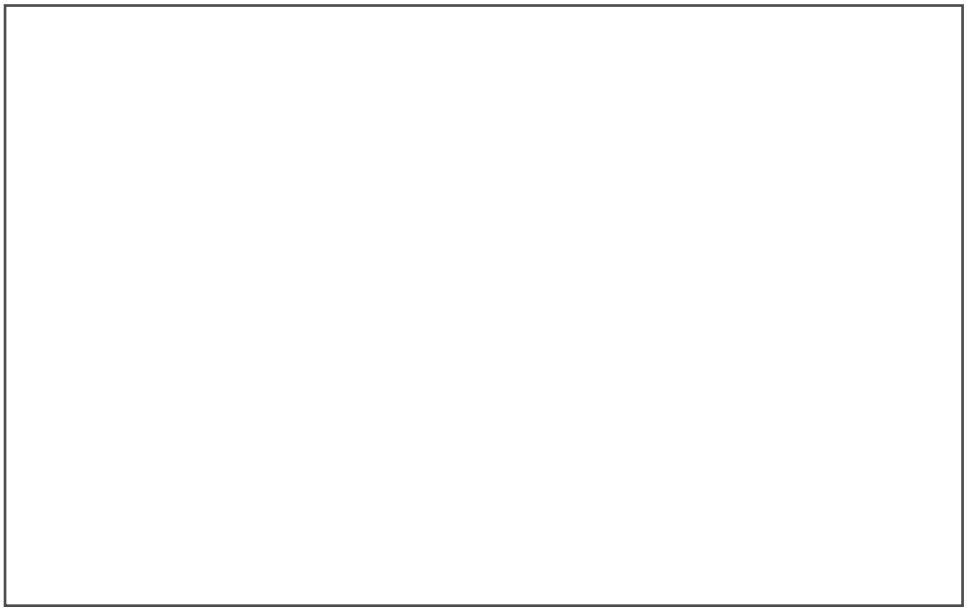
172-175 Declareer de benodigde lokale variabelen.
177-178 Roep de functie ZoekOp aan en zet het resultaat in het veld POSITIE.
179-196 Als POSITIE groter is dan -1, veeg dan regel 12 van het invoerscherm schoon. Maak vervolgens een volledige naam vanuit het te muteren record. Maak het record leeg en roep de procedure LeesNummer aan. Open TelFile en schrijf het record terug op de plaats die in POSITIE vermeld staat.

```
197-199Als POSITIE de waarde -1 bevat, roep dan de procedure
          Boodschap aan.
```
201-249Procedure Menu.

[4] 202-205Declareer de benodigde lokale variabelen.

```
   [5]207      Verbind de variabele TELFILE met het bestand
KLAPPER.DAT.
   [6]208-212Ga naar de functie Bestaat. Als deze functie False
          retourneert, maak dan een bestand KLAPPER.DAT op schijf
          aan. Sluit de file TELFILE.
   [7]214-218Maak een rood venster met gele letters op een zwarte
          achtergrond.
   [8]219-248Ga een REPEAT-lus in die pas beëindigd wordt als een
          9 ingetoetst wordt.
   [8]220-240Zet de diverse keuzemogelijkheden op de juiste
          plaats in het venster.
   [8]241      Lees een waarde in de variabele KEUZE.
   [8]246-251Bepaal met CASE KEUZE OF of de waarde die ingelezen
          is in de variabele KEUZE, aanleiding is om naar een
          procedure te springen.
```
250-252Hoofdprogramma.

[3]251 Roep de procedure Menu aan.



Afbeelding 10

Toelichting:

[1] Om telefoonnummers te kunnen verwerken, hebben we een record nodig waarin de velden Naam en Nummer kunnen worden opgeborgen. Het type TTelefoon bevat de nodige velden.

[2]Het type TTelFile wordt gedeclareerd als een file van het type TTelefoon. Dit type file verwerkt records van het type TTelefoon.

[3]Het hoofdprogramma van FILES_1 bestaat uit slechts één regel. Het roept de procedure Menu aan.

[4]In Menu aangekomen worden er lokale variabelen gedeclareerd:

TELFILE: TTelFile; File van het type TTelefoon;
KEUZE: Char; Om een keuze te laten maken;
X,Y: Byte; Om posities op het scherm aan te geven.

[5] Met Assign wordt vervolgens TELFILE verbonden met het bestand KLAPPER.DAT.

[6]Daarna moet onderzocht worden of het bestand al bestaat, of dat er een nieuw bestand op schijf aangemaakt moet worden. Hiervoor wordt de functie Bestaat aangeroepen. De functie Bestaat krijgt als parameter de naam van het bestand mee. Er wordt gebruik gemaakt van een ongetypeerde file. Deze file wordt verbonden met het bestand. In ons geval is dat dus KLAPPER.DAT.

Met de optie {$I-} wordt de onderbreking van een programma bij een optredende fout uitgezet. De file wordt met Reset geopend en met Close weer gesloten. Daarna wordt met {$I+} de onderbreking bij foutmeldingen weer aangezet. Als het bestand niet op de schijf stond, dan heeft zich een fout voorgedaan. In dat geval is het DOS-foutnummer in de Turbo Pascal-variabele IOResult gezet. Als zich geen fouten hebben voorgedaan, staat IOResult op 0. De functie Bestaat wordt dus True als IOResult een 0 bevat, anders wordt er False geretourneerd.

Als bij terugkomst in de procedure Menu geconstateerd wordt dat de functie Bestaat False retourneert, dan wordt er met Rewrite een nieuw bestand gemaakt. Omdat dit bestand elders in het programma geopend wordt, moet het nieuwe bestand onmiddellijk gesloten worden.

[7] [7]Om de gegevens van een telefoonklapper op het scherm te zetten, hebben we niet het hele scherm van de monitor nodig. Daarom wordt hier met TextBackground(0) en een ClrScr-opdracht een zwart scherm gemaakt. Met Window maken we vervolgens midden op het scherm een rechthoekje. Dit rechthoekje zal nu functioneren als monitorscherm. We zetten dus de achtergrondkleur op rood en de tekstkleur op geel.

[8]We gaan nu een REPEAT-lus in en vegen eerst het zojuist vervaardigde venster schoon. X krijgt een vaste waarde, in dit geval 5. Y krijgt de waarde 2. Deze coördinaten worden gebruikt

om de regels op de juiste plek in het venster te plaatsen.

Telkens als we een nieuwe regel maken, wordt Y met de Turbo Pascal-procedure Inc verhoogd. Als de regels op het scherm staan, lezen we met behulp van ReadKey een waarde in. Als je ReadKey gebruikt, dan hoef je je keuze niet met een druk op de Enter-toets te bevestigen. De waarde van de toets wordt rechtstreeks in KEUZE gezet. KEUZE is een Char-variabele, omdat ReadKey alleen Char-variabelen direct inleest.

Als KEUZE een waarde heeft, dan kunnen we met behulp van CASE KEUZE OF nagaan of er naar een andere procedure gesprongen moet worden. Bij de cijfers 1 tot en met 4 is dit het geval. Wordt er een ander teken ingetoetst, dan wordt de REPEAT-lus nog een keer uitgevoerd. Alleen als er een 9 is ingetoetst, wordt de lus niet opnieuw uitgevoerd. In dat geval wordt de lus verlaten.

[9]We kijken nu naar de achtereenvolgende keuzen. In VoerNummerIn wordt een lokale variabele TELEFOON van het type TTelefoon gedeclareerd. TelFile wordt met Reset geopend, en de velden van de variabele TELEFOON worden met behulp van FillChar op 0 gezet.

[10]De procedure LeesNummer wordt aangeroepen en de lokale variabele TELEFOON wordt als VAR-parameter meegegeven. In LeesNummer wordt een REPEAT-lus ingegaan.

[11]Eerst wordt nu de procedure Telefoonscherm aangeroepen. Deze procedure maakt op dezelfde wijze als in de procedure Menu een venstertje, bestemd voor het inlezen van telefoonnummers.

[12]Als dit venstertje gemaakt is, keren we terug in de procedure LeesNummer. Nu wordt achter de verschillende regels die door de procedure Telefoonscherm zijn gemaakt, een waarde gelezen in de verschillende velden van de parameter Telefoon.

[13]Als het laatste veld ingevoerd is, moeten we kijken of er wel een naam en een nummer zijn ingevoerd. Als dat in orde is, krijgen NAAM_OK en NUMMER_OK beide de waarde True. Als er iets ontbreekt, wordt de betreffende variabele False. Vervolgens wordt er gekeken of er een boodschap verzonden moet worden, en zo ja welke. Als een van de Booleans False is, dan wordt de REPEAT-lus opnieuw uitgevoerd.

[14]Als alles in orde is, keren we terug in de procedure VoerNummerIn. Omdat het hier gaat om het invoeren van een nieuw nummer, wordt de bestandswijzer met behulp van Seek aan het einde van het bestand gezet. Daarna wordt het record met Write in het bestand geschreven en wordt de file gesloten. We keren nu weer terug naar het menu.

[15]Hier aangekomen doen we net of we voor "bladeren" kozen en gaan naar de procedure Bladeren. Deze procedure geeft een voorbeeld van sequentieel lezen. De file wordt geopend. Daardoor staat de bestandswijzer op positie 0. Vervolgens gaan we een WHILE-lus in, die aangehouden wordt tot het einde van het bestand bereikt is, of tot het moment dat de variabele CH de ASCII-waarde 27 bevat. Zodra CH een ASCII-waarde 27 heeft, wil dat zeggen dat er op de Escape-toets gedrukt is. De functie EOF onderzoekt of we het einde van het bestand bereikt hebben. Na het inlezen van het eerste record wordt de bestandswijzer automatisch één record opgeschoven. De procedure Telefoonscherm wordt aangeroepen om een scherm met de juiste titel te tekenen. Het zojuist ingelezen record wordt op dit scherm afgebeeld.

Onderaan dit scherm zetten we nog de mededeling dat we met ESC terug kunnen keren naar het menu. De uitvoering wordt nu gestopt, omdat er gewacht wordt op een toetsaanslag die de variabele CH van een waarde zal voorzien. Aan het einde van de lus gaan we terug naar het begin van de WHILE-lus, waar het volgende record gelezen kan worden, of van waaruit we terugkeren naar het menu.

[16] Terug in het menu kijken we wat er gebeurt als we voor "zoeken" kiezen. In dit geval gaan we naar de procedure Zoeken. Dit is een kleine procedure die de functie ZoekOp aanroept en een boodschap afgeeft als de functie -1 retourneert.

[17]We gaan nu naar de functie ZoekOp. Hier wordt eerst een venstertje gemaakt waar de gezochte naam ingelezen kan worden. Nadat het veld Naam ingelezen is, wordt de functie Hoofdletters aangeroepen om de ingelezen naam om te zetten in hoofdletters. Dit moet gebeuren omdat bij het vergelijken van namen een naam in kleine letters een andere waarde heeft dan een naam in hoofdletters.

In een vergelijking worden Jansen en JANSEN niet als dezelfde naam gezien, omdat de ASCII-waarden verschillen. Daarom wordt in de functie Hoofdletters iedere letter van de ingelezen naam met behulp van de functie Upcase omgezet in een hoofdletter. Een kleine letter die naar Upcase gezonden wordt, wordt veranderd in een hoofdletter, terwijl een naar Upcase gezonden hoofdletter gewoon een hoofdletter blijft.

Nadat Naam is omgezet in hoofdletters, wordt de file met Reset geopend en gaan we een WHILE-lus in om het bestand sequentieel te lezen. Bij ieder ingelezen record wordt het veld Telefoon.Naam vergeleken met de ingelezen variabele Naam. Om die vergelijking gelijkwaardig te maken, wordt ook voor het veld Telefoon.Naam de functie Hoofdletters aangeroepen. Daarna wordt gekeken of een woord ter grootte van de ingelezen naam gelijk is aan het veld Telefoon.Naam. Hiervoor worden met de procedure Copy net zoveel letters van het veld Telefoon.Naam gebruikt als de opgegeven naam bevat.

[17]Als er gezocht wordt naar "Jansen" en de gebruiker tikt

"Jan" in, dan worden de drie letters van Jan vergeleken met de eerste drie letters van het veld Telefoon.Naam. Bij het record van Jansen aangekomen, wordt het inlezen gestaakt en wordt aan de gebruiker gevraagd of dit het gezochte nummer is. Als er op de "j" of "J" gedrukt wordt, dan wordt de WHILE-lus verlaten. Als in de variabele CH een "j" of een "J" staat, dan krijgt de functie ZoekOp de positie van de bestandswijzer min 1. Deze laatste correctie is noodzakelijk, omdat na het inlezen van het gezochte record de bestandswijzer met 1 verhoogd is. Als er op een andere toets dan "j" of "J" of op de Escape-toets gedrukt is, dan wordt het zoeken in het bestand voortgezet. Als de gezochte naam niet gevonden wordt, dan staat er dus geen "J" of "j" in de variabele CH en retourneert de functie ZoekOp de waarde -1. We keren dan via de procedure Zoeken terug naar het menu.

[17]De laatste keuzemogelijkheid is "mutatie". In Mutatie wordt eerst de functie ZoekOp aangeroepen om het te muteren record op te zoeken. Als het record gevonden is, wordt door de functie ZoekOp de plaats van dit record in het bestand geretourneerd. Als het record niet gevonden is, retourneert de functie ZoekOp de waarde -1. Als in de variabele POSITIE dus een waarde komt te staan die groter is dan -1, dan is het record gevonden.

In het record Telefoon staan de waarden die we willen muteren. Vanuit deze velden worden de voornaam, een eventueel tussenvoegsel en de achternaam samengevoegd tot één volledige naam. Deze naam wordt doorgegeven aan de procedure LeesNummer en in de titel opgenomen. Ook het gevonden record wordt aan deze procedure doorgegeven.

Als uit LeesNummer teruggekeerd wordt, moet het veranderde record weer op zijn oude plaats worden teruggeschreven. Hiervoor openen we de file TelFile. Met Seek zetten we de bestandswijzer op de waarde die in de variabele POSITIE staat. Met Write wordt het record teruggeschreven en met Close wordt TelFile weer gesloten.

8.12 Voorbeeld ongetypeerde file

Een ongetypeerde file kan gebruikt worden om gegevens te lezen en te schrijven in instelbare gegevensblokken. In het volgende voorbeeld worden de getallen 11 tot en met 30 in een bestand geschreven. Hiervoor wordt het type Byte gebruikt. Daarna worden de weggeschreven getallen als een type Word uit ditzelfde bestand gelezen. Met andere woorden: er worden gegevens in de vorm van blokken van één byte in het bestand geschreven. Het bestand wordt weer uitgelezen in de vorm van blokken van twee bytes. Uiteraard resulteert dit in verschillende waarden. Alleen bij ongetypeerde

files is de grootte van de elementen instelbaar. In dat geval kan aan Rewrite en aan Reset een parameter meegegeven worden waarmee de grootte van de elementen ingesteld kan worden. Als deze parameter weggelaten wordt, dan stelt Turbo Pascal een blokgrootte van 128 bytes in. Ongetypeerde files worden als blokken uitgelezen en weggeschreven. Hiervoor worden de functies BlockRead en BlockWrite gebruikt.

```
PROGRAM FILES_2;
USES CRT;

CONST
    GETALLEN:Array[1..20]  OF   Byte   =   (11,12,13,14,15,16,17,
           18,19,20,21,22,23,24,25,26,27,28,29,30);

VAR
   F       : File;
   BYTES   : Array[1..20] OF Byte;
   WOORDEN : Array[1..10] OF Word;
   I       : Byte;
   GELEZEN : Word;

BEGIN
   Assign(F,'TEST.DAT');
   Rewrite(F,1);
   BlockWrite(F,Getallen,20);
   Close(F);
   ClrScr;
   Reset(F,1);
   BlockRead(F,BYTES,FileSize(F),GELEZEN);
   ClrScr;
   Writeln('Uitdraai array BYTES :');
   FOR I := 1 TO FileSize(F) DO Writeln(BYTES[I]);
   Writeln('Gelezen elementen :',GELEZEN);
   Close(F);
   Readln;
   Reset(F,2);
   BlockRead(F,WOORDEN,FileSize(F),GELEZEN);
   ClrScr;
   Writeln('Uitdraai array WOORDEN :');
   FOR I := 1 TO FileSize(F) DO Writeln(WOORDEN[I]);
   Writeln('Gelezen elementen :',GELEZEN);
   Seek(F,GELEZEN DIV 2);
   Truncate(F);
   Close(F);
   Readln
END.
```

```
|Uitdraai array BYTES:     |Uitdraai array WOORDEN:      |
|                          |                             |
|11                        |3083                         |
|12                        |3597                         |
|13                        |4111                         |
|14                        |4625                         |
|15                        |5139                         |
|16                        |5653                         |
|17                        |6167                         |
|18                        |6681                         |
|19                        |7195                         |
|20                        |7709                         |
|21                        |                             |
|22                        |Gelezen elementen: 10        |
|23                        |                             |
|24                        |                             |
|25                        |                             |
|26                        |                             |
|27                        |                             |
|28                        |                             |
|29                        |                             |
|30                        |                             |
|                          |                             |
|Gelezen elementen: 20     |                             |
```

Regels:Toelichting:

[1]3-5Declareer een getypeerde constante GETALLEN.
[2]6-11 Declareer de benodigde variabelen.

[3]13 Verbind de file-variabele F met het bestand TEST.DAT.
[3]14Maak een nieuw bestand op schijf aan. Zet de elementgrootte op 1.
[4]15 Schrijf de constante GETALLEN in het bestand TEST.DAT.
16 Sluit de file F.
[5]18 Open de file F met een elementgrootte van 1.
[5]19 Lees de inhoud van TEST.DAT in de variabele BYTES.
[6]21-25Schrijf de inhoud van de variabele BYTES naar het scherm en laat zien hoeveel elementen er gelezen zijn. Sluit vervolgens de file F.
[7]26 Open de file F met een elementgrootte van 2.
[8]27 Lees de inhoud van TEST.DAT in de variabele WOORDEN.
29-31 Laat de inhoud van WOORDEN op het scherm zien.
[9]32-33 Verwijder het tweede deel van de file F.
34-35Sluit de file F.

Toelichting:

[1]Als eerste wordt er in het programma een getypeerde constante gemaakt in de vorm van een array met 20 elementen van het type Byte.

[2]De variabele F is een ongetypeerde file en de variabelen BYTES en WOORDEN zijn arrays van respectievelijk het type Byte en Word. De variabele BYTES heeft 20 elementen en de variabele WOORDEN heeft er 10. Gemeten in bytes zijn beide variabelen even groot. De variabele I wordt als index gebruikt en GELEZEN is bestemd om als parameter dienst te doen in BlockWrite.

[3]Nadat F met het bestand TEST.DAT verbonden is, wordt er met Rewrite een nieuw bestand op schijf gezet. Omdat de variabele F een ongetypeerde file is, kan aan Rewrite een parameter meegegeven worden. Met deze parameter wordt ingesteld hoe groot de elementen zijn die in het bestand geschreven moeten worden. Omdat het hier gaat om gegevens van het type Byte, wordt de elementgrootte ingesteld op 1.

[4]Bij de Turbo Pascal-procedure BlockWrite moet je als parameter meegegeven: de file waar naartoe geschreven moet worden, de buffer waaruit gelezen moet worden en het aantal elementen dat uit de buffer naar het bestand geschreven moet worden. De array GETALLEN kunnen we zien als een buffer. Met deze opdracht worden er dus 20 elementen uit GETALLEN naar TEST.DAT geschreven. Omdat GETALLEN 20 elementen groot is, wordt dus de hele array weggeschreven. Na het wegschrijven kan de file gesloten worden.

[5]Om te laten zien dat er inderdaad 20 elementen van één byte groot in het bestand geschreven zijn, wordt de file weer geopend met een elementgrootte van 1. Met BlockRead wordt het bestand TEST.DAT in de variabele BYTES gelezen. BlockRead krijgt de volgende parameters mee: de file waaruit gelezen moet worden, de buffer waar naar geschreven moet worden en het aantal elementen dat gelezen moet worden. Bovendien wordt een variabele als VAR-parameter meegestuurd, in dit geval de variabele GELEZEN. In deze variabele houdt BlockRead bij hoeveel elementen er daadwerkelijk gelezen zijn. Stel dat de buffer die ingelezen moet worden te klein is om het hele bestand in één keer te bevatten. In dat geval kan de aanroep van BlockRead herhaald worden tot GELEZEN op 0 staat.

De Turbo Pascal-functie FileSize wordt gebruikt om de grootte van het bestand aan te geven. Omdat de file geopend is met een elementgrootte van één byte, geeft FileSize een bestandsgrootte terug van 20.

[6]Als de inhoud van de variabele BYTES naar het scherm

geschreven wordt, dan zien we dat de array BYTES een kopie is van de array GETALLEN.

[7]Nu openen we de file F met een elementgrootte van 2. Met BlockRead zetten we de inhoud van TEST.DAT in de variabele WOORDEN.

[8]De getallen die nu op het scherm getoond worden, hebben een andere waarde dan de getallen die in de variabele BYTES staan. Dit komt doordat het bestand nu uitgelezen is met een elementgrootte van 2. Bij de variabele BYTES werden er groepjes van 8 bits ingelezen, terwijl de grootte van de elementen in WOORDEN 16 bits is. FileSize geeft nu een bestandsgrootte van 10 aan. Dat wil zeggen: 10 elementen van 2 bytes.

[9]Met "Seek(GELEZEN DIV 2)" wijst de bestandswijzer in het midden van het bestand. Als Truncate aangeroepen wordt, dan wordt datgene wat na de bestandswijzer staat uit het bestand verwijderd. In dit geval wordt het bestand dus gehalveerd. Truncate werkt niet op tekstfiles.

8.13 Voorbeeld tekstfile

Tekstfiles worden in Turbo Pascal anders behandeld dan andere files. In andere files kunnen getallen opgeslagen worden met op het eerste gezicht nietszeggende bitpatronen. We spreken dan over binaire files. In tekstfiles daarentegen staan in principe alleen maar tekens die zo naar de printer of het scherm geschreven kunnen worden. In binaire files staat van tevoren vast hoe groot de elementen zullen zijn, in tekstfiles niet. Iedere zin of elk woord dat in het bestand geschreven wordt, kan immers een andere lengte hebben. In het programma FILES_3 kan er tekst in bestanden geschreven worden. Deze tekst wordt vervolgens random uitgelezen en samengevoegd tot zinnen.

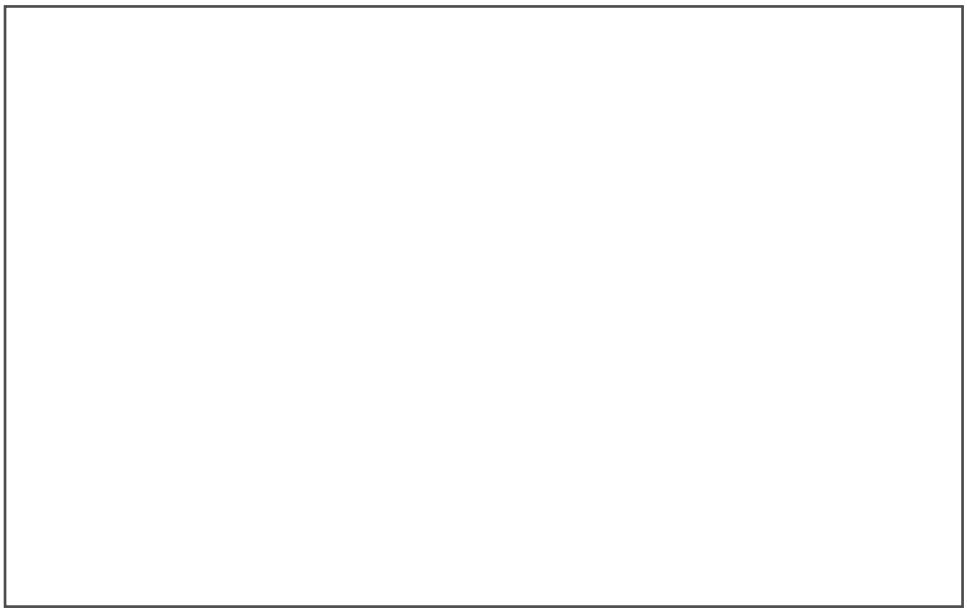
Met "random uitlezen" wordt hier bedoeld dat het programma bepaalt welke elementen gelezen zullen worden. Op deze manier ontstaan er willekeurige zinnen. Het volgende programma zal het een en ander verduidelijken:

```
PROGRAM FILES_3;
USES CRT;

CONST
   ESC = #27;

VAR
   BIJV, ZELFST  : Word;
   FBIJV, FZELFST: Text;

FUNCTION Bestaat(FileNaam:String):Boolean;
VAR
   F: File;
BEGIN
   Assign(F,FileNaam);
   {$I-}
   Reset(F);
   Close(F);
   {$I+}
   Bestaat := IOResult = 0
END;

FUNCTION ZinNummer(VAR Bestand:Text;NR:Word):String;
VAR
   IND: Word;
   ZIN: String;
BEGIN
   Reset(Bestand);
   IND := 0;
   ZIN := '';
   WHILE IND <= NR DO
   BEGIN
      Readln(Bestand,Zin);
      Inc(IND)
   END;
   Close(Bestand);
   ZinNummer := ZIN
END;

PROCEDURE Zinnen;
VAR
   ZIN   : String;
   NUMMER: Word;
   LETTER: Char;
   I     : Byte;
BEGIN
```

```
   REPEAT
      ClrScr;
      GotoXY(15,2);
      Write('AFDRUKKEN ZINNEN');
      ZIN := '';
      FOR I := 1 TO 2 DO
      BEGIN
         NUMMER := Random(BIJV);
         ZIN := ZIN + ZinNummer(FBIJV,NUMMER) + ' ';
         NUMMER := Random(ZELFST);
         ZIN := ZIN + ZinNummer(FZELFST,NUMMER) + ' ';
         IF I = 1 THEN Zin := Zin + 'voor '
      END;
      GotoXY(5,5);
      Write(Zin);
      GotoXY(5,8);
      Write('Wilt U nog meer zinnen zien?  (j/n)');
      LETTER := ReadKey;
   UNTIL LETTER IN ['N','n']
END;

PROCEDURE Bestanden;
BEGIN
   BIJV := 0;
   ZELFST := 0;
   Assign(FBIJV,'BIJV.TXT');
   IF NOT Bestaat('BIJV.TXT') THEN
   Rewrite(FBIJV) ELSE
   BEGIN
      Reset(FBIJV);
      WHILE NOT EOF(FBIJV) DO
      BEGIN
         Readln(FBIJV);
         Inc(BIJV)
      END;
      Close(FBIJV)
   END;
   ZELFST := 0;
   Assign(FZELFST,'ZELFST.TXT');
   IF NOT Bestaat('ZELFST.TXT') THEN
   Rewrite(FZELFST) ELSE
   BEGIN
      Reset(FZELFST);
      WHILE NOT EOF(FZELFST) DO
      BEGIN
         Readln(FZELFST);
```

```
         Inc(ZELFST)
      END;
      Close(FZELFST)
   END
END;

PROCEDURE Invoeren(VAR Bestand:Text;Titel:String;
                   VAR Aantal:Word);
VAR
   WOORD: String;
BEGIN
   ClrScr;
   GotoXY(10,2);
   Write(Titel);
   Append(Bestand);
   GotoXY(5,4);
   Write('Voer in... ');
   Readln(WOORD);
   IF WOORD > '' THEN
   BEGIN
      Writeln(Bestand,WOORD);
      Inc(Aantal)
   END;
   Close(Bestand)
END;

PROCEDURE Menu;
VAR
   KEUZE: Char;
BEGIN
   KEUZE := #0;
   Bestanden;

   REPEAT
      ClrScr;
      GotoXY(20,2);
      Write('M E N U ');
      GotoXY(10,5);
      Write
      (' 1  =  Invoeren bijvoeglijk naamwoorden.');
      GotoXY(10,6);
      Write
      (' 2  =  Invoeren zelfstandig naamwoorden.');
      GotoXY(10,7);
      Write(' 3  =  Lezen zinnen');
      GotoXY(10,9);
```

```
      Write('       Escape = Einde');
      KEUZE := ReadKey;
      CASE KEUZE OF
      '1' :Invoeren
      (FBIJV,'INVOER BIJVOEGLIJK NAAMWOORDEN',BIJV);
      '2' :Invoeren
      (FZELFST,'INVOER ZELFST. NAAMWOORDEN',ZELFST);
      '3' :Zinnen;
       ESC:Exit
      END;
   UNTIL KEUZE = ESC
END;

BEGIN
   TextBackground(0);
   ClrScr;
   TextBackground(7);
   TextColor(0);
   Window(10,8,75,17);
   Randomize;
   Menu
END.
```



Afbeelding 11

```
Regels:Toelichting:

   [1]3-4Declareer de constante ESC.
   [2]5-7Declareer de benodigde variabelen.
```
[6]8-18Function Bestaat(FileNaam:String):Boolean;
11-18Onderzoek of het bestand FileNaam op de schijf staat.

[11]19-34Function ZinNummer(VAR Bestand:Text;Nr:Word):String;
23-34 Lees de tekst die aangegeven wordt door NR uit het bestand.
35-61Procedure Zinnen.
42-60Een REPEAT-lus die verlaten wordt als er geen zinnen meer gemaakt hoeven te worden.

```
46        Maak de lokale variabele ZIN leeg.
47-54Loop de FOR-lus twee keer door.
  [10]49  Geef de random-waarde door aan NUMMER.  De  waarde  mag
          niet groter zijn dan de waarde die in BIJV staat.
  [12]50  Zet de tekst uit de file FBIJV, aangegeven door NUMMER,
          in de variabele ZIN.
51Geef NUMMER een nieuwe random-waarde. De waarde mag niet groter
          zijn dan de waarde die in ZELFST staat.
52Zet de tekst uit de file FZELFST, aangegeven door NUMMER, in de
          variabele ZIN.
56        Zet de samengestelde zin op het scherm.
57-60Vraag of er nog meer zinnen gemaakt moeten worden.  Als  het
          antwoord "n" of "N" is, verlaat  dan  de  lus  en  keer
          terug naar het menu.
```
[5]62-91 Procedure Bestanden.

64-65 Zet BIJV en ZELFST op 0.

[7]66-77Verbind FBIJV met het bestand BIJV.TXT. Als dit bestand niet bestaat, maak het dan aan. Als het wel bestaat, open het dan en lees het aantal regels dat het bestand bevat. Zet dit aantal in BIJV.

[8]78-90Bepaal het aantal regels in ZELFST.TXT. Als het bestand niet bestaat, maak het dan aan.

[9]92-110Procedure Invoeren(VAR Bestand:Text;Titel:String;
VAR Aantal:Word)

97-103Maak een scherm met het opschrift zoals ontvangen is in de parameter Titel. Open de file, aangegeven door de parameter Bestand, om toe te kunnen voegen. Lees een waarde in de variabele WOORD.

104-109Als de lokale variabele WOORD een tekst bevat, schrijf het dan in het bestand en tel 1 op bij de VAR-parameter Aantal. Hiermee wordt BIJV of ZELFST bijgewerkt. Sluit de file.

111-141Procedure Menu.

[5] 116Roep de procedure Bestanden aan.

117-140Maak een keuze mogelijk voor het invoeren van zelfstandig naamwoorden, bijvoeglijk naamwoorden en voor het uitlezen van zinnen.

142-150Hoofdprogramma.

[3]143-147Maak een lichtgrijs venstertje op een zwarte achtergrond met zwarte letters.

[4] 148Activeer de random-generator.

149 Ga naar de procedure Menu.

Toelichting:

[1]Het programma gebruikt een constante ESC die de waarde heeft van teken 27 uit de ASCII-tabel. Het toetsenbord voert dit teken in als de Escape-toets wordt ingedrukt.

[2]Het programma moet weten hoeveel tekstelementen er in BIJV.TXT staan en hoeveel in ZELFST.TXT. Daarvoor worden de variabelen BIJV en ZELFST gebruikt. Deze variabelen zijn van het type Word. Om een verbinding te leggen met de tekstbestanden, worden de variabelen FBIJV en FZELFST gebruikt. Deze variabelen zijn van het type Text. Het type Text is een voorgedefinieerd file-type in Turbo Pascal. Text is gedefinieerd als File OF Char.

[3]In het hoofdprogramma wordt op de bekende manier midden op het scherm een venstertje gemaakt. In dit geval is het venster lichtgrijs met zwarte letters.

[4]Met het aanroepen van Randomize wordt de random-generator ingesteld. Random moet ons willekeurige getallen geven. Deze getallen zijn echter schijnbaar willekeurig. Random berekent ze aan de hand van een beginwaarde. Randomize ontleent deze beginwaarde aan de klok van de computer.

[5]In Menu wordt eerst de procedure Bestanden aangeroepen. In deze procedure wordt FBIJV verbonden met BIJV.TXT en wordt de variabele BIJV op 0 gezet. FZELFST wordt met ZELFST.TXT verbonden en ZELFST wordt op 0 gezet. Om te zien of het bestand reeds op de schijf staat, wordt de functie Bestaat aangeroepen. De bestandsnaam wordt als parameter meegegeven.

[6]In de functie Bestaat wordt gebruik gemaakt van een ongetypeerde file, die gedeclareerd is als lokale variabele. Deze file wordt verbonden met de parameter FileNaam. Vervolgens wordt de compileerinstructie {$I-} geplaatst. Als deze instructie geplaatst is, zal het programma niet onderbroken worden als er een lees- of schrijffout op schijf optreedt.

We openen nu met Reset de file. Als het bestand niet op schijf gevonden wordt, dan wordt de Turbo Pascal-variabele IOResult gevuld met de waarde van de optredende DOS-fout. Als IOResult de waarde 0 oplevert, is er geen fout opgetreden en is het bestand op de schijf gevonden. De compileer-instructie {$I+} zorgt ervoor dat het programma weer zal worden onderbroken als er een fout optreedt.

[7]Als de functie Bestaat een False retourneert, dan wordt met Rewrite een nieuw bestand op schijf aangemaakt. Als het bestand wel bestaat, dan wordt het met Reset geopend en gaan we een WHILE-lus in. In deze lus wordt met Readln telkens een tekstregel ingelezen. Readln weet wanneer een tekstregel afgelopen is. Iedere regel wordt namelijk afgesloten met een teken 13 en 10 uit de ASCII-tabel. De tekens 13 en 10 worden ook wel CR en LF genoemd. CR staat voor "carriage return" (wagen terug) en LF voor "line feed" (regel opschuiven). Readln zoekt in een tekst naar #13 en #10 en weet dat dit het einde van de regel is. Writeln voegt bij het schrijven in een tekstbestand deze tekens automatisch toe aan de weg te schrijven tekstregel.

[8]Na het lezen van de regel word BIJV opgehoogd. Als het hele bestand gelezen is, bevat BIJV het aantal regels in het bestand. Eenzelfde werkwijze wordt gevolgd met het bestand ZELFST.TXT. Het aantal regels wordt in ZELFST gezet.

[9]Teruggekeerd in het menu kiezen we voor het invoeren van een bijvoeglijk naamwoord. We gaan naar de procedure Invoeren en hebben FBIJV als parameter meegekregen, evenals een tekst voor het invoerscherm. Als we een zelfstandig naamwoord hadden gekozen, dan waren we naar dezelfde procedure gegaan, maar met andere parameters. In de procedure Invoeren wordt het bestand geopend met Append. Append wordt bij tekstfiles gebruikt om regels tekst aan een bestand toe te voegen.

Daarna wordt vanaf het toetsenbord een waarde ingelezen in de

lokale variabele WOORD. Als er niets in WOORD staat, dan wordt de procedure verlaten. Heeft deze variabele wel een waarde, dan wordt de tekst met behulp van Writeln aan het bestand toegevoegd. Writeln voegt, zoals gezegd, tevens de tekens 13 en 10 uit de ASCII-tabel aan de tekstregel toe.

De VAR-parameter AANTAL is verbonden met de velden ZELFST of BIJV. Als er een woord wordt toegevoegd aan het bestand, dan worden de velden waarmee de parameter verbonden is met 1 verhoogd.

[10]We zijn weer terug in het menu en kiezen voor Zinnen. Deze keuze brengt ons in de procedure Zinnen. Hier krijgt eerst de lokale variabele Nummer een waarde. Hiervoor roepen we de functie Random aan. Deze functie retourneert een pseudo-willekeurige waarde. Omdat we BIJV als parameter meegeven, zal de door Random geretourneerde waarde niet hoger zijn dan de waarde die in BIJV staat. In BIJV staat het aantal regels dat het bestand BIJV.TXT bevat.

[11]De nu volgende FOR-lus wordt twee maal uitgevoerd. Eerst wordt ZIN gevuld met de waarde die de functie ZinNummer retourneert. ZinNummer krijgt FBIJV en het zojuist door Random gegenereerde nummer als parameters mee.

ZinNummer opent de parameter Bestand en voert een WHILE-lus uit waarin telkens een regel tekst uit het bestand gelezen wordt. Iedere keer dat er een regel gelezen is, wordt IND verhoogd. Als IND dezelfde waarde heeft als de parameter NR, dan wordt de lus verlaten. In de lokale variabele ZIN staat nu de gezochte tekst. Het bestand wordt gesloten en de functie retourneert de gezochte tekst.

[12]Teruggekeerd in de procedure Zinnen, wordt de door ZinNummer geretourneerde tekst toegevoegd aan de lokale variabele ZIN. Dan wordt ZinNummer aangeroepen om een tekst uit het bestand ZELFST.TXT te lezen, aangegeven door de variabele NUMMER die van Random opnieuw een waarde heeft gekregen. Als I de waarde 1 heeft, wordt het woord "voor" in ZIN gezet.

We keren terug in de FOR-lus en lezen nogmaals twee teksten in. ZIN heeft nu achtereenvolgens een bijvoeglijk naamwoord, een zelfstandig naamword, het woordje ’voor’ en opnieuw een bijvoeglijk naamwoord en een zelfstandig naamwoord gelezen. Het resultaat is een willekeurig gevormde zin, die op het scherm wordt gezet.

8.14 Nog wat zaken over files

Voor iedere file die in een programma gedeclareerd wordt, houdt Turbo Pascal een record bij. Dit record, dat bij de file hoort, kun je lezen en veranderen. Het programma FILES_4 laat zien hoe dit in z'n werk gaat:

```
PROGRAM FILES_4;
USES CRT, DOS;

VAR
   F      : File;
   TF     : Text;
   LETTER : Char;

BEGIN
   ClrScr;
   Assign(F,'BESTAND.DAT');
   Rewrite(F);
   WITH FileRec(F) DO
   BEGIN
      Write('Handle  = ',Handle);
      Write(' De mode = ',Mode);
      Writeln(' De Naam = ',Name)
   END;
   Assign(TF,'TEKST.TXT');
   Rewrite(TF);
   WITH TextRec(TF) DO
   BEGIN
      Write('Handle  = ',Handle);
      Write(' De mode = ',Mode);
      Writeln(' De Naam = ',Name)
   END;
   Writeln(TF,'Dit is de eerste regel tekst.');
   Writeln(TF,'We doen er nog een tweede regel bij.');
   Writeln (TF,'Tenslotte nog een derde regel.');
   Flush(TF);
   Reset(TF);

   WHILE SeekEOF(TF) <> TRUE DO
   BEGIN
      Read(TF,LETTER);
      Write(Letter);
      IF SeekEoLn(TF) THEN Writeln
   END;
   Readln;
   Close(F);
   Close(TF)
END.
```

Regels: Toelichting:

3-6Declareer een ongetypeerde file, een textfile en een Char-variabele.
```
 9-10Maak een bestand BESTAND.DAT aan en verbind het met F.
   [1]11-16Laat enkele gegevens uit het gegevensrecord van F
          zien.
17-18Maak een bestand TEKST.TXT aan en verbind het met FT.
   [2]19-24    Laat enkele gegevens uit het gegevensrecord van FT
          zien.
25-27Schrijf drie regels tekst in het bestand TEKST.TXT.
   [3]28       Maak de buffer van FT leeg.
   [4]30-35Lees het tekstbestand uit met behulp van SeekEOF en
          SeekEoLn.
37-38Sluit de files.
```

Toelichting:

```
   [1]Als de file F verbonden is met het bestand BESTAND.DAT,
kunnen we in het gegevensrecord van F kijken naar de gegevens die
daar opgeslagen zijn. Het record dat door Turbo Pascal gebruikt
wordt, is bestemd voor getypeerde- en ongetypeerde files, maar
niet voor
```
TextFiles. De manier waarop het record hier gelezen wordt met de opdracht FileRec(F), wordt "typecasting" genoemd. Dit houdt in dat je naar F kijkt alsof het een variabele van het type FileRec is. Dit type heeft de volgende vorm:

FileRec = Record
```
   Handle: Word;
        Mode: Word;
        RecSize: Word;
        Private: Array[1..26] OF Byte;
        UserData: Array[1..16] OF Byte;
        Name: Array[0..79] OF Char;
     END;
```

In het veld Handle staat een nummer dat ontvangen wordt van het besturingssysteem, de zogenaamde file handler. In het veld Mode staat of de file wel of niet open is en zo ja, op welke manier de file gebruikt mag worden. In Turbo Pascal zijn enkele velden gedeclareerd die gebruikt kunnen worden om de status van de file te controleren of te veranderen:

```
|Constante:       |Waarde:         |Betekenis:                   |
|FmClosed         |55216           |File gesloten.               |
|FmInput          |55217           |File open voor invoer.       |
|FmOutput         |55218           |File open voor uitvoer.      |
|FmInOut          |55219           |File open voor in- en uitvoer. |
```

```
Bij een getypeerde file staat in
```
RecSize de afmeting van het record dat gebruikt

wordt. Bij een ongetypeerde file is dit 128 bytes, behalve als met Reset of Rewrite een andere afmeting opgegeven wordt. In de arrays Private en UserData worden gegevens voor het functioneren van de file bijgehouden. Het veld Name bevat de naam van het bestand, waar de file aan gekoppeld is.

[2]Voor een file van het type Text wordt door Turbo Pascal een ander type record gebruikt. Dit record, dat op dezelfde manier bereikt kan worden als een record van het type FileRec, is wat uitgebreider. Het heeft de volgende vorm:

```
TextRec = Record
   Handle: Word;
        Mode: Word;
        BufSize: Word;
        Private: Word;
        BufPos: Word;
        BufEnd: Word;
        BufPtr: PTextBuf;
        OpenFunc: Pointer;
        InOutFunc: Pointer;
        FlushFunc: Pointer;
        CloseFunc: Pointer;
        UserData: Array[1..16] OF Byte;
        Name: Array[0..79] OF Char;
        Buffer: TTextBuf;
     END;
```

De velden Handle, Mode, Private, UserData en Name dienen hetzelfde doel als de overeenkomstige velden in het type FileRec.

Het veld BufPtr wijst naar een Array[0..127] OF Char. In Turbo Pascal is zo'n array gedefinieerd als type TextBuf. Het veld BufSize heeft dan de waarde 128. Je kunt BufPtr naar een zelfgedefinieerde grotere buffer laten wijzen. Je moet dan het veld BufSize aanpassen. De velden BufPos en BufEnd worden gebruikt om met de buffer te kunnen werken. De velden OpenFunc, InOutFunc, FlushFunc en CloseFunc wijzen naar interne Turbo Pascal-procedures om files te openen, te bewerken, de buffer leeg te maken en de files te sluiten. Hiervoor in de plaats kan de programmeur eigen procedures en functies maken waar deze velden naar kunnen wijzen.

[3]De drie regels die in het bestand worden geschreven, worden bij een schrijfopdracht eerst in de buffer geplaatst. Als de buffer vol is, wordt er pas daadwerkelijk naar de schijf geschreven. Het aanroepen van Flush zorgt er bij textfiles voor, dat de inhoud van de buffer naar schijf geschreven wordt. Als je met Close een file sluit, dan roept Close altijd de Turbo Pascal-procedure Flush aan.

[4]De Turbo Pascal-functie SeekEOF wordt op dezelde manier gebruikt als de functie EOF. Het verschil met EOF is, dat SeekEOF de spaties, tabs en de markering voor het einde van een regel overslaat bij het lezen van een tekstbestand. De functie SeekEoLn wordt True als het einde van een tekstregel gelezen wordt. Omdat in dit geval het bestand letter voor letter uitgelezen wordt en SeekEOF de

markering voor het einde van een regel overslaat, wordt SeekEoLn gebruikt om naar een volgende regel te springen. SeekEoLn werkt volgens hetzelfde principe als EoLn, alleen leest EoLn alle tekens (inclusief de spaties en de tabs), terwijl SeekEoLn alleen de tekens leest en de spaties en de tabs overslaat.

8.15 Apparaatfiles

Dit hoofdstuk over files zou niet compleet zijn als we het niet zouden hebben over apparaatfiles. We hebben gezien hoe een file het transport van gegevens verzorgt tussen het geheugen van de computer en een gegevensbestand op de schijf. Bij het verkeer tussen het geheugen en bijvoorbeeld de printer is er ook sprake van een gegevensstroom. In dit geval vanuit het geheugen naar de printer. Ook dit verkeer wordt door het type File verzorgd. Een aantal van dit soort files heeft in Turbo Pascal al een naam gekregen. Dit zijn overigens dezelfde namen als de namen die het besturingssysteem gebruikt:

CON staat voor een verbinding met het scherm van de monitor voor uitvoer, en voor invoer via het toetsenbord.

LPT1, LPT2, LPT3 staan voor een verbinding met drie mogelijke printers. De printerfiles kunnen alleen voor uitvoer gebruikt worden.

COM1, COM2, AUX staan voor de seriële poorten COM1 en COM2. AUX kan in plaats van COM1 gebruikt worden. Seriële poorten worden onder andere gebruikt voor verbinding met een modem of een muis.

NUL staat voor een file waarnaar niet kan worden weggeschreven. Zo'n file kun je gebruiken als het programma een filenaam vereist terwijl je geen file kunt gebruiken.

Als bij Read of Write geen file als parameter wordt doorgegeven, dan wordt zondermeer CON aangenomen. Het volgende programmaatje FILES_5 schrijft de tekst naar het scherm zoals Write dat gedaan zou hebben in geval er geen file genoemd zou zijn:

```
PROGRAM FILES_5;
VAR
   FCON: Text;

BEGIN
   Assign(FCON,'CON');
   Append(FCON);
   Writeln(FCON,'Dit komt op het scherm.');
   Readln
END.
```

De variabele FCON is een file van het type Text. Het type Text is een File OF Char. FCON wordt verbonden met het bestand CON. De schrijfopdracht Writeln stuurt de tekst nu gewoon naar het beeldscherm.

In Turbo Pascal is de file-variabele LST in de unit PRINTER voorgedefinieerd. Als je in je programma de opdracht "USES PRINTER" opneemt, dan kun je LST gebruiken. We kunnen echter ook zelf een variabele maken die naar de printer schrijft. Het programmaatje FILES_6 laat zien hoe dat in z'n werk gaat:

```
PROGRAM FILES_6;
VAR
   FPRINT: Text;

BEGIN
   Assign(FPRINT,'LPT1');
   Append(FPRINT);
   Writeln(FPRINT,'Dit gaat naar de printer.'#12);
END.
```

De variabele FPRINT van het type Text wordt verbonden met de LPT1-poort en stuurt de tekst naar de printer. De toevoeging #12 heeft betrekking op ASCII-waarde 12, ofwel FF (van FormFeed). Deze zorgt ervoor dat het papier van de printer doorgeschoven wordt.

Voor het schrijven van tekst naar de printer hoeven we echter niet zo moeilijk te doen. Het volgende programmaatje laat zien hoe het makkelijker kan:

```
PROGRAM FILES_7;
USES PRINTER;

BEGIN
   Writeln(LST,'Dit gaat naar de printer.'#12)
END.
```

Je ziet dat als we de file-variabele LST gebruiken, we met minder regels hetzelfde resultaat krijgen.

8.16 Opgaven

8.1 Schrijf een programma dat in tekstbestanden die op de schijf staan voor alle letters van het alfabet telt hoe vaak ze in de tekst voorkomen. In het programma moet je een bestand kunnen opgeven; het programma moet vervolgens het bestand openen, de letters tellen en een rapport met de getelde letters op het scherm tonen. Als het opgegeven bestand niet op de schijf staat, moet dit gemeld worden.

```
--------------------
121

          121

          121

160

          160

160
```