In dit boek is getracht je op een logische manier kennis te laten maken met Turbo Pascal en de mogelijkheden die deze taal biedt. Van allerlei faciliteiten zijn voorbeeldprogramma's gemaakt en toegelicht. Als je dan voor je gevoel helemaal door de taal heen gegaan bent, kom je op het laatst tot de ontdekking dat er toch nog zaken zijn die je niet behandeld hebt. Vandaar de titel van dit laatste hoofdstuk.

20.1 Lussen onderbreken

In de diverse voorbeelden heb je een flink aantal keren de FOR-, de WHILE- en de REPEAT-lus zien opduiken. In alle voorbeelden werden de lussen volledig afgewerkt. Zodoende is de mogelijkheid om een deel van de lus over te slaan of voortijdig te verlaten niet behandeld. Turbo Pascal beschikt met Break en Continue over een tweetal procedures die een FOR-, een WHILE- en een REPEAT-lus kunnen reguleren. Het programma LUS_1 demonstreert dit:

```
PROGRAM LUS_1;
USES CRT;
VAR
   I: Byte;

BEGIN
   ClrScr;
   FOR I := 1 TO 100 DO
   BEGIN
      IF I IN [11..30] THEN Continue;
      IF I = 80 THEN Break;
      Write(I:3);
      IF I MOD 20 = 0 THEN Writeln;
   END;
   Writeln;
   Write('Buiten de lus. I = ',I);
   ReadKey
END.
```

Als LUS_1 gedraaid wordt, dan zie je dat de getallen 11 tot en met 30 niet op het scherm verschijnen, en dat er gemeld wordt dat I de waarde 80 heeft op het moment dat de lus verlaten wordt. De FOR-lus in het programma LUS_1 zou normaal 100 keer uitgevoerd worden. Bij iedere doorloop van de lus wordt I met 1 verhoogd.

In het programma is opgenomen:

IF I IN [11..30] THEN Continue;

Deze opdracht zorgt ervoor dat als de waarde van I in het bereik van [11..30] ligt, de lus niet verder uitgevoerd wordt, maar dat er gesprongen wordt naar de END die de lus afsluit. De Write-opdracht wordt dan overgeslagen. Verder staat er:

IF I = 80 THEN Break;

Deze opdracht vertelt dat als I de waarde 80 heeft, de lus verlaten moet worden. Normaal zou de lus 100 maal uitgevoerd worden. Door de opdracht Break wordt de lus echter al verlaten als I gelijk is aan 80.

20.2 Geluid produceren

We kunnen met Turbo Pascal via het luidsprekertje van de computer geluid produceren. De commando's Sound en NoSound uit de unit CRT zetten het geluid respectievelijk "aan" en "uit". Het programma GELUID laat zien hoe ze gebruikt worden:

```
PROGRAM GELUID;
USES CRT;
CONST
   LAAG = 1000;
   HOOG = 2000;
VAR
   I: Word;

BEGIN
   FOR I := LAAG TO HOOG DO
   BEGIN
      Sound(I);
      Delay(10)
   END;
   FOR I := HOOG DOWNTO LAAG DO
   BEGIN
      Sound(I);
      Delay(10)
   END;
   NoSound
END.
```

De constanten HOOG en LAAG worden gebruikt om in een FOR-lus de waarde van de variabele I te regelen. De variabele I begint bij LAAG en wordt bij iedere doorloop met 1 opgehoogd, net zolang tot I

gelijk is aan HOOG. I wordt als parameter meegegeven aan Sound. De waarde in I staat voor het aantal Herz van het geluidssignaal. Bij het draaien van het programma krijgen we een pieptoon te horen, die van laag naar hoog gaat. In de tweede lus gaat de waarde van I van HOOG naar LAAG en krijgen we dus een pieptoon van hoog naar laag te horen. Als Sound aangeroepen wordt, blijft de luidspreker geluid produceren tot NoSound het geluid weer afzet.

20.3 DOS-gegevens

Een programma werkt nauw samen met het besturingssysteem MS-DOS. Het kan van belang zijn dat je gegevens waar het besturingssysteem mee werkt, vanuit je programma kunt onderzoeken en eventueel aanpassen. De unit DOS levert hiervoor een aantal mogelijkheden. Het programma DOS_1 laat achtereenvolgens zien onder welke DOS-versie het programma draait, welke omgevingsvariabelen gebruikt worden en wat de status van de Verify- en Ctrl-Break-vlag is:

```
PROGRAM DOS_1;
USES CRT, DOS;
VAR
   BREAK, VERIFY: Boolean;

BEGIN
   ClrScr;
   Writeln('De versie van DOS op dit systeem      =  ',
            Lo(DosVersion),'.',Hi(DosVersion));
   Writeln('Het aantal strings in DOS-environment = ',
            EnvCount);
   Writeln('Het ingestelde path in DOS            = ',
            GetEnv('PATH'));
   Writeln('De directory van COMMAND.COM          = ',
            GetEnv('COMSPEC'));
   GetCBreak(BREAK);
   Writeln('De stand van Ctrl-Break               = ',
            BREAK);
   SetVerifY(True);
   GetVerify(VERIFY);
   Writeln('De stand van de Verify-vlag           = ',
            VERIFY);
  ReadKey
END.
```

Regels: **Toelichting:**

```
3-4Declareer variabelen voor het gebruik met GetCBreak en
        GetVerify.
   [1]7-8Laat het nummer van de DOS-versie zien.
   [2]9-10Laat zien hoeveel strings er in de DOS-omgeving staan.
    [3]11-14Toon het in DOS ingestelde zoekpad en laat de
        directory zien waar COMMAND.COM staat.
   [4]15-17Onderzoek de status van de Ctrl-Break-vlag. Toon het
        resultaat op het scherm.
   [5]18-21Zet de Verify-vlag op True en laat zien dat dit
        inderdaad gebeurd is.
```

Toelichting:

[1]Iedere keer dat er een nieuwe DOS-versie uitkomt, krijgt deze versie een nieuw nummer. Het versienummer bestaat uit twee getallen die van elkaar gescheiden worden door een punt. Als er kleine veranderingen in een versie aangebracht worden, wordt het getal achter de punt verhoogd. Als de veranderingen groot zijn, wordt het getal vóór de punt verhoogd. Zo is het versienummer DOS 3.2 dus de derde DOS-versie waar voor de tweede keer veranderingen in zijn aangebracht.

DOS is "upward compatible". Dat wil zeggen dat programma's die onder een oude DOS-versie gecompileerd zijn, ook draaien op een nieuwere versie van het besturingssysteem. Dit wil natuurlijk niet zeggen dat programma's die onder een nieuwe versie zijn gecompileerd, ook vlekkeloos zullen draaien onder een oudere versie van DOS. Daarom is het van belang dat je het versienummer van het besturingssysteem waaronder je programma draait, kunt vergelijken met de versie van het besturingssysteem waaronder het gecompileerd is. Mocht dit een ouder systeem zijn, dan kun je de uitvoering van het programma stoppen en de boodschap naar het scherm sturen dat het programma een nieuwere versie van het besturingssysteem nodig heeft.

De procedure DosVersion retourneert een type Word. In de lage byte staat het versienummer. In de hoge byte staat de aanvulling. Om de bytes van elkaar te scheiden, kun je de Turbo Pascal-functies Lo en Hi gebruiken.

[2]De DOS-omgeving bestaat uit een aantal strings. In deze strings kan een aantal zaken opgegeven worden: het zoekpad waar DOS mee moet werken, de manier waarop de prompt afgebeeld moet worden, waar COMMAND.COM staat, enzovoort. De functie EnvCount retourneert het aantal strings dat de DOS-omgeving bevat.

[3]De variabelen PATH en COMSPEC komen bij de meeste systemen voor. Vandaar dat we met behulp van het commando GetEnv de waarde van beide variabelen in het besturingssysteem naar het scherm sturen. GetEnv retourneert een string met de waarde van de gevraagde variabele en krijgt de naam van de variabele als parameter door (kijk voor de benaming van de diverse variabelen in je DOS-

handleiding).

[4]Als Ctrl-Break wordt ingedrukt, wordt er door het besturingssysteem een vlag gezet. Tijdens in- en uitvoerbewerkingen controleert DOS of de Ctrl-Break-vlag gezet is. Als dit het geval is, wordt het programma onderbroken. Met SetCBreak kun je vanuit Turbo Pascal de Ctrl-Break-vlag op True of False zetten. De status van de vlag kun je uitlezen met GetCBreak.

[5]DOS kan controleren of de uitvoer naar een schijf goed verlopen is. Mogelijk is een schijf beschadigd. Met de Verify-vlag aan, wordt een beschadiging op de schijf geconstateerd. Als deze vlag uit staat, blijft de controle achterwege. Vanuit een Turbo Pascal-programma kun je de status van de Verify-vlag regelen: met GetVerify kun je de status controleren, en met SetVerify kun je de status op True of False zetten.

20.4 Omleiding

Turbo Pascal heeft de variabele ExitProc. Deze variabele is van het type Pointer en bevat het adres van de Exit-procedure. De Exit-procedure beëindigt de uitvoering van het programma en herstelt het geheugen in de situatie zoals die was bij de aanvang van het programma. Of een programma nu gewoon beëindigd wordt, of geforceerd met behulp van Halt, of door een onderbreking vanwege een fout, in alle gevallen wordt de procedure Exit aangeroepen.

Als we nu het adres in ExitProc vervangen door het adres van een zelfgemaakte procedure en in die procedure ExitProc weer zijn oude waarde geven, dan zal onze eigen procedure tijdens de beëindiging van een programma altijd aangeroepen worden. Je kunt dan nog allerlei zaken regelen. Je kunt hierbij denken aan het bijwerken van bestanden, het legen van buffers en het sluiten van files. Het programma EXIT_1 demonstreert het gebruik van de variabele ExitCode:

```
PROGRAM EXIT_1;
USES CRT;
VAR
   OUDE_PROC: Pointer;
PROCEDURE NieuweExitProcedure;FAR;
BEGIN
   ExitProc := OUDE_PROC;
   Writeln
   (’Hier kan nog van alles afgesloten worden’);
   CASE ExitCode OF
      0  : Writeln(’Programma normaal beëindigd.’) ;
      500: Writeln(’Programma met Halt beëindigd.’);
   END;
   ReadKey
END;
```

```
BEGIN
   ClrScr;
   OUDE_PROC := ExitProc;
   ExitProc  := @NieuweExitProcedure;
   Randomize;
   IF Random(2) = 1 THEN Halt(500)
END.
```

Bij de start van het programma wordt het adres dat in ExitProc staat, bewaard in het veld OUDE_PROC. In ExitProc wordt het adres van de procedure NieuweExitProcedure gezet. Als het programma beëindigd wordt, dan wordt er een sprong uitgevoerd naar het adres dat in ExitProc staat. Omdat ExitProc het adres bevat van de procedure NieuweExitProcedure, komen we na beëindiging van het programma dus in de zelfgeschreven procedure terecht.

In een dergelijke procedure zou je nog allerlei zaken kunnen regelen alvorens het programma definitief te beëindigen. Nu is er alleen maar een Writeln-opdracht in gezet om te laten zien dat de procedure ook werkelijk aangeroepen wordt. Bij de procedure NieuweExitProcedure zijn twee zaken van groot belang. Ten eerste moet een dergelijke procedure altijd de toevoeging FAR hebben. Als je dit vergeet, kan het programma vastlopen. Ten tweede moet op de eerste regel de variabele ExitProc altijd weer zijn oorspronkelijke waarde krijgen.

De manier waarop het programma beëindigd wordt, is afhankelijk van het getal dat toegewezen wordt door de functie Random. Als Random een 1 genereert, dan wordt het programma beëindigd met een aanroep van Halt. Als het gegenereerde getal ongelijk is aan 1, dan wordt het programma normaal beëindigd.

In de procedure NieuweExitProcedure wordt de waarde van de Turbo Pascal-variabele ExitCode onderzocht. Als deze variabele de waarde 0 heeft, dan is het programma normaal beëindigd. Als het programma beëindigd wordt door een foutmelding, dan bevat ExitCode het nummer van de geconstateerde fout. ExitCode kan ook de waarde bevatten van een aan Halt meegegeven parameter. Als deze parameter buiten het bereik van de foutcodes ligt kan ErrorCode nooit fout geïnterpreteerd worden.

20.5 Parameters op de commandoregel

Om een programma op te starten, geef je achter de DOS-prompt de naam van het programma op en, als alles in orde is, gaat het programma vervolgens draaien. Turbo Pascal kent de mogelijkheid om meteen tijdens de start aan het programma gegevens te verstrekken. Deze gegevens worden dan achter de naam van het programma ingetoetst. Het programma kan nu kijken of er gegevens verstrekt zijn, en zo ja, met deze gegevens iets doen.

Het programma TEL_OP telt twee getallen bij elkaar op die bij de start met het commando werden meegegeven. Het programma kijkt of het aantal parameters klopt. Als dit niet het geval is, wordt daar melding van gemaakt. Als het aantal parameters in orde is, dan wordt gekeken of er twee getallen zijn ingevoerd. Als er iets anders dan getallen is ingevoerd, dan maakt het programma daar melding van. Als alles in orde is, wordt de optelsom op het scherm gezet.

Als je het programma in de ontwikkelomgeving draait, kun je de parameters zetten door te kiezen voor «*Run*» en «*Parameters*»:

```
PROGRAM TEL_OP;
USES CRT, STRINGS;

VAR
   I: Byte;
   GETAL_1, GETAL_2: Integer;

   FUNCTION ZetOm(NR:Byte;VAR GETAL:Integer): Boolean;
   VAR
      CODE: Integer;

   BEGIN
      Val(ParamStr(NR),GETAL,CODE);
      ZetOm := CODE = 0
   END;



BEGIN
   ClrScr;
   CASE ParamCount OF
      0: Writeln('Geef twee getallen op');
      2: BEGIN
      IF ZetOm(1,GETAL_1) AND ZetOm(2,GETAL_2) THEN
      Write('De uitkomst van ',GETAL_1:5, ' + ',
             GETAL_2:5, ' = ', Longint(GETAL_1) + GETAL_2)
      ELSE
         Write('U moet getallen opgeven.')
      END;
      ELSE Writeln('Niet meer dan twee getallen S.v.p.')
   END;
   ReadKey
END.
```

Regels:Toelichting:

3-5Declareer de benodigde globale variabelen.

6-12Function ZetOm(NR:Byte,VAR GETAL:Integer):Boolean.

[2]10-11Converteer ParamStr(NR) naar een integer en plaats deze in de parameter GETAL. Als de conversie goed verlopen is, dan retourneert ZetOm de waarde True, anders wordt de waarde False geretourneerd.

[1]15-25Voer een CASE OF-constructie uit met ParamCount. Controleer of er twee parameters zijn ingevoerd. Als dit het geval is, controleer dan of het getallen zijn en zet de optelling op het scherm.

[2]18Roep ZetOm aan om de strings om te laten zetten in een getal.

[3]20-22Als ZetOm de waarde True retourneert, toon dan de uitkomst op het scherm. Als ZetOm de waarde False retourneert, vraag dan om invoer van getallen.

Toelichting:

[1]De CASE OF-constructie in het begin van het programma wordt uitgevoerd met de Turbo Pascal-variabele ParamCount. In deze variabele staat het aantal parameters dat is meegegeven vanaf de commandoregel. Als ParamCount een waarde heeft van 0, of groter is dan 2, zijn er dus geen of teveel parameters doorgegeven. Dat wordt gemeld en het programma wordt beëindigd. Als er twee parameters zijn doorgegeven, wil dat nog niet zeggen dat het ook getallen zijn.

[2]De functie ZetOm krijgt als parameter een nummer mee en de parameter GETAL is een VAR-parameter van het type Integer. Het nummer wordt gebruikt om de juiste ParamStr uit te lezen. ParamStr is een functie in Turbo Pascal die een op de commandoregel ingevoerde parameter in de vorm van een string retourneert. De parameter NR geeft aan welk gegeven uit de lijst van parameters bedoeld wordt.

Val wordt gebruikt om de string om te zetten in een integer. Als de conversie mislukt, is CODE ongelijk aan 0. ZetOm retourneert dan False.

[3]Als ZetOm twee maal True retourneert, zijn er twee getallen ingevoerd en kan de som met de uitkomst op het scherm worden gezet. Let nog even op de gebruikte typecasting:

Longint(GETAL_1)

Deze typecasting maakt een veld van het type Longint. Hier wordt de variabele GETAL_2 bij opgeteld. Als dit achterwege zou blijven, zou de uitkomst nooit hoger kunnen zijn dan 32757, het grootste positieve getal dat in een Integer past.

20.6 Het starten van een ander programma

Je kunt vanuit je Turbo Pascal-programma een ander programma starten en na beëindiging daarvan weer terugkeren in je programma. Dit is handig als je programma even de diensten van een ander programma nodig heeft. Je kunt hierbij denken aan het starten van een tekstverwerker of een editor omdat je een stukje tekst moet maken of lezen. Ook als je een menu-programma met Turbo Pascal wilt maken, heb je deze faciliteit nodig.

Het programma START activeert de commandovertaler COMMAND.COM. We verhuizen dan tijdelijk naar DOS. Als je in DOS het commando "Exit" geeft, kom je weer terug in het programma:

```
{$M 4000,0,0}
PROGRAM START;
USES DOS, CRT;

VAR
   COMMANDO: ComStr;

BEGIN
   COMMANDO := GetEnv('COMSPEC');
   SwapVectors;
   Exec(COMMANDO,'');
   SwapVectors
END.
```

Met GetEnv('COMSPEC') wordt het pad voor COMMAND.COM opgehaald. Dit pad wordt in het veld COMMANDO gezet. Voordat we het programma kunnen verlaten, moeten eerst de adressen in de interrupttabel weer in orde gemaakt worden. De interrupttabel is in hoofdstuk 15 behandeld. Een aantal adressen in de interrupttabel wordt bij de start van een programma vervangen. Deze adressen worden bewaard in de SaveIntXX-variabelen. Dit zijn variabelen die voorgedefinieerd zijn in Turbo Pascal. De "XX" staat hier voor het interruptnummer. De procedure SwapVectors verwisselt de adressen die in de interrupttabel staan met de adressen in de SaveIntXX-variabelen.

De procedure Exec start onmiddellijk daarna het programma. Exec krijgt twee parameters mee. De eerste bevat het volledige pad (inclusief de programmanaam) van het te starten programma, de tweede bevat de parameters voor het programma.

Bij terugkeer uit het gestarte programma wordt de oude situatie in de interrupttabel hersteld door opnieuw SwapVectors aan te roepen.

Het starten van andere programma's kan geheugenproblemen veroorzaken. Als de maximale omvang van de heap op 655360 gezet is, zal de compiler alle resterende geheugenruimte aan de heap toewijzen. Als je het programma start, wordt dus de volledige geheugenruimte gebruikt. Als je dan een ander programma wilt starten lukt dit niet, omdat alle geheugenruimte al een bestemming heeft. Bij gebruik van Exec moet je ervoor zorgen dat er voldoende ruimte in het geheugen overblijft om het programma te kunnen bevatten. Dit kun je bereiken door met de $M-compileroptie een andere geheugenindeling in te stellen (zie paragraaf 19.8).

20.7 Teksten naar de printer sturen

Het doorgeven van teksten aan een printer kan nog wel eens gecompliceerd zijn en aanleiding geven tot hoofdbrekens. Een printer is in staat om opdrachten te ontvangen van de computer waarop het is aangesloten. Het gecompliceerde is nu dat de printeropdrachten per type printer verschillen. Er bestaat hiervoor geen echte standaard. Wel is het zo dat fabrikanten van printers zich vaak conformeren aan de opdrachtenset van een groot printermerk. Dat neemt niet weg dat de opdrachtensets van IBM, Epson en Hewlett Packard veel van elkaar verschillen. Deze verschillen worden door software-ontwikkelaars overbrugt door zogenaamde printerdrivers. Dit is software die printercommando's die in een programma opgenomen zijn, vertaalt naar printercommando's die de printer begrijpt. Een printerdriver kan door de gebruiker in een programma worden ingelezen. Zodoende kan het programma printercommando's aanmaken die speciaal bestemd zijn voor de aangesloten printer.

In programma's die je schrijft, moet je dus rekening houden met de printer die je gebruikt. Het voorbeeldprogramma PRTEKST zal niet bij alle printers het gewenste resultaat hebben. In dit programma wordt de printertaal PCL van Hewlett Packard gebruikt. Als je geen Hewlett Packard-printer hebt, moet je de corresponderende commando's van jouw printer opzoeken in de printerhandleiding die bij je printer is meegeleverd.

De printer is in Turbo Pascal gedefinieerd als file van het type

Text. Deze file is verbonden met het bestand "LPT1" (zie paragraaf 8.15, "Apparaatfiles"). De naam van deze file is in de Turbo Pascal-unit PRINTER gedefinieerd als LST. De commando's Write en Writeln nemen, als er geen filenaam wordt doorgegeven, aan dat de file CON bedoeld wordt. De file CON is verbonden met het beeldscherm. Om teksten ergens anders heen te leiden is het noodzakelijk om het bestand op te geven waar de teksten heen moeten. Een commando voor de printer in Turbo Pascal begint met de filenaam LST:

Write(LST,.....

Na de komma achter LST volgt dan een aantal lettertekens die naar de printer gestuurd moet worden. Deze lettertekens kunnen samen zowel een regel tekst als een commando voor de printer vormen. Met deze printercommando's kun je aan de printer de papierlengte, het aantal regels, de regelafstand, het te gebruiken lettertype, enzovoort opgeven.

Je kunt je nu afvragen hoe de printer commando's en tekst uit elkaar kan houden. Een commando voor de printer begint altijd met het letterteken 27 (#27) uit de ASCII-tabel. Dit teken duidt een ESC aan. Na het ESC-teken volgt een combinatie van lettertekens en nummers die een bepaalde opdracht aan de printer vormt. Deze combinaties kun je vinden in de handleiding die je bij je printer hebt gekregen.
In PCL laat de volgende combinatie aan de printer weten dat er A4-papier gebruikt zal worden:

Write(LST,#27'&''l''2''6''A');

De opdracht:

Write(LST'Een regel tekst');

laat de printer de doorgezonden regel afdrukken.

```
PROGRAM PRTEKST;
USES CRT, PRINTER;

BEGIN
   Write(LST,#27,'E');
   Write(LST,#27,'&','l','1','H');
   Write(LST,#27,'&','2','6','A');
   Write(LST,#27,'&','l','2','D');
   Write(LST,#27,'&','l',2,'E');
   Write(LST,#27,'&','a',10,'L');
   Write(LST,'Eerste regel onder de bovenmarge ');
   Writeln(LST,'met een kantlijn van 10.');
   Writeln(LST,'Ga over op cursief drukken.');
   Write(LST,#27,'(','s','1','S');
   Writeln(LST,'Nu naar Times en onderstreept.');
   Write(LST,#27,'(','s','0','S');
   Write(LST,#27,'(','s','4','1','0','1','T');
   Write(LST,#27,'&','d','0','D');
   Writeln(LST,'Dit is lettertype Times onderstreept.');
   Write(LST,#27,'&','d','@');
   Write(LST,#27,'(','s','4','0','9','9','T');
   Write(LST,'Onderstreping uit en lettertype Courier');
   Writeln(LST,#12)
END.
```

Regels: Toelichting:

```
4          Reset de printer.
5          Zet de papierinvoer op lade.
6          Zet de pagina-afmeting op A4.
7          Zet de regelafstand op 2 regels per inch.
8          Zet de bovenmarge op 2 regels.
9          Zet de linkermarge op 10 posities.
10-12           Stuur regels tekst naar de printer.
13         Schakel over op cursief drukken.
14         Stuur tekst naar de printer.
15         Zet rechtop drukken aan.
16         Schakel over op het lettertype Times.
17         Zet onderstrepen tekst aan.
18         Stuur tekst naar de printer.
19         Zet onderstrepen uit.
20         Schakel over op het lettertype Courier.
22         Werp de pagina uit.
```

In het programma PRTEKST worden Write en Writeln door elkaar heen gebruikt. Het verschil tussen deze twee is dat Writeln aan de doorgezonden tekst de "einde regel"-tekens toevoegt en dat Write dit niet doet. Als we vóór het printercommando Writeln zouden opnemen, dan zouden we telkens ongewild een regel opschuiven. In de laatste regel zie je dat letterteken 12 gebruikt wordt. Dit letterteken veroorzaakt een zogenaamde Form Feed, ofwel het uitwerpen van de pagina. Merk op dat hierbij niet de filenaam LST gebruikt wordt.

401