5. Vergelijkingen

5.1 IF...THEN...ELSE

Tijdens de uitvoering van een computerprogramma worden telkens vergelijkingen gemaakt. De uitkomst van een vergelijking bepaalt het verdere verloop van de uitvoering.

Stel, we willen een programma maken dat uit een lijst van persoonsgegevens degenen moet selecteren die morgen jarig zijn. Als iemand morgen jarig is, krijgt hij of zij een felicitatiebrief. Het programma moet de volgende vergelijking maken:

Is deze persoon morgen jarig?

Zo ja: Ga dan naar dat gedeelte van het programma waar een felicitatiebrief geschreven wordt en schrijf de brief.

In Free Pascal zou zo'n vergelijking er zo uit kunnen zien:

```
IF GeboorteDatum = Datum_Vandaag + 1 THEN
         Schrijf felicitatiebrief;
```

We kunnen bij een IF THEN-constructie ook met BEGIN en END werken:

```
IF GeboorteDatum = Datum_Vandaag + 1 THEN
BEGIN
         Schrijf felicitatiebrief;
         Stuur een order naar de bloemenman;
         Maak vijfentwintig gulden over op de rekening van
       de bloemenman
END;
```

De opdrachten die tussen BEGIN en END staan worden alleen uitgevoerd als iemand morgen jarig is.

In de voorgaande voorbeeldjes zijn we er steeds van uitgegaan, dat als er niet aan de voorwaarde voldaan wordt, er niets moet gebeuren. Het is ook mogelijk dat er wel degelijk iets moet gebeuren als niet aan de voorwaarde voldaan wordt.

Een voorbeeld hiervan is de volgende instructie:

```
IF aan voorwaarde voldaan THEN
   Doe het ene
ELSE
   Doe het andere;
```

Vóór het beschermde woord ELSE wordt geen puntkomma gezet, en uiteraard is ook hier een constructie met BEGIN en END mogelijk. Nu een echt werkend programma:

```
PROGRAM VERGEL_1;
USES CRT;

VAR
```

```
   OK: Boolean;

BEGIN
   OK := True;
   ClrScr;
   IF OK THEN
   BEGIN
      GotoXY(10,4);
      Write('Nu is OK True')
   END
      ELSE
   BEGIN
      GotoXY(50,15);
      Write('Nu is OK False')
   END;
   Readln;
END.
```

Toelichting:

Het programma test of de Boolean-variabele OK de waarde True heeft. Omdat dit het geval is, verschijnt de mededeling "Nu is OK True". Als OK niet de waarde True zou hebben, zou de mededeling gedaan worden dat OK False is. Als je de regel:

```
OK := True;
```

vervangt door:

```
OK := False;
```

zal de tweede mededeling "Nu is OK False" worden afgedrukt.

In plaats van:

```
IF OK
```

kun je ook schrijven:

```
IF OK = True
```

Beide schrijfwijzen zijn correct. Voor de leesbaarheid van de programmacode is de eerste manier de beste, aangenomen dat tevens sprake is van duidelijk herkenbare namen voor variabelen. Kijk in het programma nog eens hoe de puntkomma's geplaatst zijn. In de praktijk blijken hiermee nog wel eens fouten gemaakt te worden.

Een IF THEN-constructie kan genest worden. Dit wil zeggen dat binnen een IF THEN-constructie opnieuw een IF THEN opgenomen kan worden. Het volgende programmaatje demonstreert dit:

```
PROGRAM VERGEL_2;
USES CRT;
VAR
   OK: Boolean;
   Getal: Word;

BEGIN
   OK := True;
   Getal := 8;
   ClrScr;
   IF OK THEN
   BEGIN
      GotoXY(10,4);
      Write('Nu is OK True');
      GotoXY(20,12);
      IF GETAL < 10 THEN
      Write('Het getal is kleiner dan tien')
      ELSE
      Write('Het getal is groter dan negen')
   END
```

```
      ELSE

   BEGIN
      GotoXY(50,15);
      Write('Nu is OK False')
   END;
   Readln
END.
```

Toelichting:

In het programma VERGEL_2 is er een variabele GETAL bijgekomen. Als de variabele OK de waarde True heeft, wordt een mededeling op het scherm gezet. Daarna wordt gekeken of GETAL kleiner dan 10 is. Het resultaat wordt op het scherm meegedeeld.

Als OK False is, wordt GETAL dus niet gemeten. Gebruik de interne debugger om de loop van het programma te volgen (zie hiervoor bijlage 3).

5.2 Logische AND en OR

De logische AND- en OR-operatoren worden gebruikt om verschillende condities te testen.

Dit gaat in de volgende vorm:

Als dit waar is EN dat is waar, doe dan iets.

In dit geval moeten beiden beweringen waar zijn voordat actie volgt. Dit is heel wat anders dan als je zegt:

Als dit waar is OF dat is waar, doe dan iets.

Als het zo geformuleerd staat, moet er al actie volgen als een van beide waar is. Als beide beweringen waar zijn, moet er natuurlijk ook actie volgen. Als geen van beiden waar is, gebeurt er eenvoudig niets.

In het programma VERGEL_3 worden de opdrachten na ELSE uitgevoerd als niet aan beide voorwaarden voldaan wordt:

```
PROGRAM VERGEL_3;
USES CRT;

VAR
   OK: Boolean;
   GETAL: Word;

BEGIN
   OK := True;
   GETAL := 1;
   ClrScr;
   IF OK AND (GETAL < 10) THEN
   BEGIN
      GotoXY(10,4);
      Write
      (’Nu is OK True en GETAL kleiner dan tien.’)
   END
      ELSE
   BEGIN
      GotoXY(40,15);
      Write(’OK is ’,OK);
      IF NOT (GETAL < 10) THEN
      Write(’ GETAL is groter dan negen.’)
      ELSE
      Write(’ GETAL is kleiner dan tien.’);
   END;
   Readln
END.
```

Regels: Toelichting:

3-5 Declareer de benodigde variabelen.
7-8 Zet OK op True en GETAL op 1.
9 Veeg het scherm schoon.
[1]10-15Als OK True is en GETAL is kleiner dan 10, ga dan naar de tiende positie op regel 4 en schrijf daar een mededeling.
[2]16-24Als OK False is of GETAL is groter dan 9, ga dan naar de veertigste positie op regel 15 en schrijf daar de waarde van OK. Zet op het scherm of GETAL kleiner dan 10 is of groter dan 9.

Toelichting:

[1]"IF OK" heeft hetzelfde resultaat als "IF OK = True". Als we het op deze manier zouden schrijven, zou "IF OK = True" tussen haakjes moeten staan:

```
IF (OK = True) AND (GETAL < 10)
```

Je kunt als vuistregel nemen dat de expressies aan weerszijden van de AND- of OR- operator tussen haakjes geplaatst moeten worden. Alleen als het een Boolean-variabele betreft, en er getest wordt op de manier zoals in het programma, kunnen de haakjes achterwege blijven. Met het gebruik van de AND-operator moeten beide condities waar zijn.

[2]Als de condities niet allebei waar zijn, dan is er minstens één niet waar. Een Boolean als OK kent maar twee mogelijke waarden: True en False. Met een Write-opdracht wordt deze waarde op het scherm gezet. Bij de volgende test wordt de NOT-operator gebruikt. Deze operator draait de conditie om. De opdracht:

```
IF NOT (GETAL <= 10)
```

wil zeggen: "Als GETAL groter of gelijk is aan 10". Het zal duidelijk zijn dat een getal dat groter of gelijk aan 10 is, niet kleiner dan 10 kan zijn. Met deze notering wordt dus meegedeeld dat GETAL groter dan 9 moet zijn. Als het niet groter dan 9 is, dan is het dus kleiner dan 10. Het werken met de NOT-operator kan bijzonder handig zijn en ook de leesbaarheid van een programma vergroten. De volgende opdracht is toch heel duidelijk:

```
IF NOT OK THEN ...
```

De AND- en de OR-operator kunnen door elkaar gebruikt worden, zoals we in het programma VERGEL_4 zullen zien:

```
PROGRAM VERGEL_4;
USES CRT;
VAR
   OK: Boolean;
   GETAL, GETAL_2: Word;

BEGIN
   OK := True;
   GETAL := 1;
   ClrScr;
   GotoXY(20,10);
   IF (OK AND (GETAL=1)) OR (NOT OK AND (GETAL=9))            THEN
```

```
Write('Conditie is waar')
         ELSE
         Write('Conditie is niet waar');
         Readln
      END.
```

Toelichting:

Hier is het deel met de IF THEN-constructie belangrijk. Er had ook kunnen staan:

IF ((OK = True) AND (GETAL=1)) OR

((OK = False) AND (GETAL=9))

Zoals het in het programma omschreven staat, moet de variabele OK de waarde True hebben en moet in GETAL de waarde 1 staan. Beide condities moeten waar zijn. Er is nog een andere mogelijkheid: als OK False is en er staat een 9 in GETAL, dan is het ook goed. Ook in dit geval moeten beide condities waar zijn. Let er op dat de expressies aan weerszijden van de OR-operator tussen haakjes geplaatst worden. Dit is niet altijd nodig, maar het bevordert wel de leesbaarheid.

5.3 CASE...OF

Een IF THEN-constructie kan nog wel eens knap ingewikkeld worden. Om de zaak compact en overzichtelijk te houden, hebben we binnen Free Pascal de beschikking over het CASE OF-statement. Dit wordt gebruikt om een lange rij IF THEN-statements niet uit te hoeven schrijven als we iets willen kiezen. De vorm is heel eenvoudig. Tussen de woorden CASE en OF komt de naam te staan van de variabele die getest moet worden, en daaronder de waarden die getest moeten worden:

```
CASE GETAL OF
   1    : Doe iets;
   2    : Doe iets anders;
   3..10: Doe nog iets anders;
   ELSE
   Doe iets totaal anders
END;
```

Als GETAL de waarde 1 heeft, dan wordt "Doe iets" uitgevoerd. Heeft GETAL de waarde 2, dan wordt "Doe iets anders" uitgevoerd. Zit de waarde van GETAL in 3 tot en met 10, dan wordt "Doe nog iets anders" uitgevoerd. Bevat GETAL geen van deze waarden, dan wordt "Doe iets totaal anders" uitgevoerd. Een CASE OF kan alleen uitgevoerd worden met een opsommingstype. Het werkt met het type Char, Integer, Real en Boolean, of met zelfgemaakte opsommingstypen. Met strings kan geen CASE OF worden uitgevoerd.

Het programma VERGEL_5 geeft een voorbeeld van het gebruik van een CASE OF-constructie. In dit programmaatje zijn dingen opgenomen die in de vorige hoofdstukken al aan de orde zijn gekomen:

```
PROGRAM VERGEL_5;
USES CRT;

VAR
   GETAL: Word;
   CODE: Integer;
   STR_GETAL, BOODSCHAP: String;
   LETTER: Char;

BEGIN
   TextBackground(0);
   ClrScr;
   Window(15,10,65,13);
   TextBackground(1);
   TextColor(14);
   ClrScr;
   LETTER := #0;
   WHILE NOT (LETTER IN ['N','n']) DO
   BEGIN
      REPEAT
         GotoXY(2,2);
         Write
         ('Voer een getal in tussen 1 en 10... ');
         Readln(STR_GETAL);
         Val(STR_GETAL, GETAL, CODE);
         IF CODE <> 0 THEN
         BEGIN
            ClrScr;
            GotoXY(2,2);
            Write
            ('Voor een getal cijfers gebruiken!');
            GotoXY(2,3);
            Write('Druk op Enter...');
            Readln;
            ClrScr
         END;
      UNTIL CODE = 0;
      CASE GETAL OF
         1: BOODSCHAP :=
            'U voerde het cijfer 1 in.';
         2: BOODSCHAP :=
            'U voerde het cijfer 2 in.';
         3: BOODSCHAP :=
            'U voerde het cijfer 3 in.';
         4..10: BOODSCHAP :=
            'U voerde het cijfer '+STR_GETAL+' in.';
         ELSE
         BOODSCHAP :=
         'Een getal tussen 1 en 10 invoeren S.v.p.'
      END;
      ClrScr;
```

```
      GotoXY(2,2);
      Write(BOODSCHAP);
      GotoXY(2,3);
      Write('Nog een getal invoeren? (J/N)');
      LETTER := Readkey;
      ClrScr
   END
END.
```

Regels:Toelichting:

[1]3-7Declareer de benodigde variabelen.
[2]9-14Definieer de achtergrond- en de tekstkleur, veeg het scherm schoon en maak midden op het scherm een venster.
[3]15Zet het letterteken 0 uit de ASCII-tabel in de variabele LETTER.
[4]16-56 Ga een WHILE-lus in, die duurt tot in LETTER een "N" of een "n" staat.
[5]18-35Ga een REPEAT-lus in, die duurt tot CODE een 0 bevat.
19-21Ga naar de tweede positie op de tweede regel van het actieve venster en vraag naar een getal tussen 1 en 10.
[6]22 Lees een waarde in STR_GETAL vanaf het toetsenbord.
[6]23Maak van de string STR_GETAL een echt getal (een integer) en zet dit getal in de variabele GETAL.
[6]24-34Als CODE ongelijk aan 0 is, veeg dan het venster schoon en plaats hierin de mededeling dat uitsluitend getallen ingevoerd mogen worden. Wacht op een druk op Enter en veeg het venster schoon.
[5]35 Einde van de REPEAT-lus.
[7]36-48Vul met behulp van een CASE OF de string BOODSCHAP. Zet in de boodschap welk getal ingevoerd is. Als de waarde in GETAL buiten het bereik van deze CASE OF ligt, zet dan in BOODSCHAP dat er een getal tussen 1 en 10 ingevoerd moet worden.
[8]49-52Veeg het scherm schoon en schrijf de inhoud van BOODSCHAP in het venster.
[8]53-54Vraag of er nog een getal ingevoerd moet worden en lees met behulp van Readkey een waarde in LETTER.
[4]56Einde van de WHILE-lus.

Toelichting:

[1]De variabele GETAL is bestemd om straks in de CASE OF-constructie getoetst te worden op zijn waarde. STR_GETAL wordt gebruikt om een getal van het toetsenbord te lezen. STR_GETAL wordt omgezet (geconverteerd) naar GETAL. Uit CODE kunnen we aflezen of de omzetting geslaagd is, en LETTER gebruiken we om te zien of er nog meer getallen ingevoerd moeten worden.

[2]Zoals gewoonlijk definiëren we eerst het scherm waarop de informatie wordt afgedrukt. De achtergrond wordt blauw en de tekst heldergeel. Het scherm wordt schoongemaakt en met de Free Pascal-procedure Window wordt midden op het scherm een venstertje gemaakt. Informatie die naar het scherm wordt geschreven, verschijnt nu in dit venstertje.

[3]Omdat LETTER gebruikt wordt om te zien of er nog meer getallen moeten worden ingevoerd, wordt hierin het letterteken 0 uit de ASCII-tabel gezet. Dit speciale letterteken wordt gebruikt om een variabele een neutrale waarde te geven.

[4] De variabele LETTER wordt getoetst aan het begin van de WHILE-lus. Door het gebruik van de NOT-operator wordt hier getoetst of LETTER een andere inhoud heeft dan "N" of "n". Vandaar dat het letterteken 0 in LETTER gezet is. Nul is geen "N" of "n". De lus zal dus altijd ingegaan worden. Als er niets in LETTER gezet zou zijn, zou de kans bestaan dat er toevallig een "N" of een "n" in LETTER zou staan. In dat geval zou de lus overgeslagen worden.

[5]De REPEAT-lus die we nu ingaan, moet voorkomen dat een ander gegeven dan een getal ingetoetst wordt. We blijven binnen deze lus zolang de variabele CODE een andere waarde bevat dan 0. De REPEAT-lus werkt tot aan UNTIL. Bij UNTIL wordt getest of de lus verlaten kan worden. Bij een WHILE-lus wordt aan het begin van de lus getest of aan de voorwaarde voldaan wordt om de lus in te gaan. Bij een REPEAT-

lus wordt de test aan het einde van de lus uitgevoerd. Dit heeft tot gevolg dat een REPEAT-lus altijd tenminste 1 keer uitgevoerd wordt.

[6]We gebruiken de string STR_GETAL om een waarde in te lezen. Als je rechtstreeks een waarde in GETAL zou inlezen, en de gebruiker zou niet een cijfer maar een letterteken intoetsen, dan zou de uitvoering van het programma onmiddellijk gestopt worden. Een string kan echter alle mogelijke tekens bevatten. We gebruiken nu Val om de inhoud van de string STR_GETAL om te zetten naar een numerieke waarde. De opdracht:

```
Val(STR_GETAL,GETAL,CODE);
```

zet de letters (alfanumerieke waarde) in STR_GETAL om naar een integer en plaatst deze waarde in de variabele GETAL. Als de operatie is geslaagd, heeft CODE de waarde 0. Als de operatie mislukt doordat STR_GETAL niet uitsluitend cijfers of een minteken bevat, dan staat in CODE de plaats in de string waar een ander gegeven dan een cijfer of minteken aangetroffen werd. Als CODE ongelijk aan 0 is, wordt een foutmelding naar het scherm gestuurd. Als CODE 0 is, wordt de REPEAT-lus verlaten.

[7]We komen nu bij de CASE OF-constructie. Je ziet dat tussen CASE en OF de naam van de variabele staat die getoetst moet worden. In dit geval is dat GETAL. Daarachter staat de waarde die getoetst moet worden, gevolgd door een dubbele punt. De opdracht:

```
1: BOODSCHAP := ’U voerde het cijfer 1 in’;
```

Heeft hetzelfde resultaat als:

```
IF GETAL = 1 THEN BOODSCHAP :=
         ’U voerde het cijfer 1 in’;
```

Op regel 43 staat:

4..10: BOODSCHAP := ...

Als de waarde in GETAL groter of gelijk aan 4 is en kleiner of gelijk aan 10, wordt de variabele BOODSCHAP gevuld. Je kunt hier zien hoeveel schrijfwerk een CASE OF-constructie bespaart. In een IF THEN-constructie zou deze opdracht als volgt geformuleerd moeten worden:

```
IF (GETAL >= 4) AND (GETAL <= 10) THEN BOODSCHAP :=
```

Als geen van de te testen waarden in GETAL voorkomt, worden de opdrachten na ELSE uitgevoerd. De variabele GETAL moet dan een waarde bevatten die kleiner dan 1 of groter dan 10 is. Er wordt dus gevraagd om een getal tussen 1 en 10.

[8]De inhoud van BOODSCHAP wordt nu naar het scherm gestuurd. Onmiddellijk daarna volgt de vraag of er nog een getal ingevoerd moet worden. Met behulp van de Free Pascal-functie Readkey wordt een waarde in LETTER gelezen. Zolang de variabele LETTER niet de waarde "N" of "n" bevat, blijven we binnen de WHILE-lus die op regel 16 begon. Readkey leest een letter van het toetsenbord zonder dat op de Enter-toets gedrukt hoeft te worden.

5.4 Opgaven

5.1Maak een programma dat om de invoer van een string vraagt. Als de string leeg is omdat de gebruiker alleen maar de Enter-toets heeft ingedrukt, moet het programma hiervan melding maken en het programma beëindigen. Als de string niet leeg is, moet het programma nagaan hoe vaak een bepaalde

letter in de string voorkomt. Het programma geeft daarna zijn bevindingen weer. Maak voor de boodschappen venstertjes.
