13. Expressies en operatoren

13.1 Inleiding

Bij het programmeren moeten we veelvuldig gebruik maken van expressies. Het volgende sommetje en de volgende bewering zijn bijvoorbeeld expressies:

25 + 30 = 55
25 < 30

Het plusteken in de som en het kleiner-dan-teken in de bewering zijn zogenaamde operatoren. De getallen aan weerszijden heten de operanden. Expressies zijn combinaties van operanden en operatoren, die gezamenlijk een waarde representeren. Het sommetje is een rekenkundige expressie, die als uitkomst de waarde 55 heeft. De vergelijking is een logische expressie die als uitkomst de waarde True of False heeft. Turbo Pascal heeft operatoren voor rekenkundige en logische expressies, maar ook voor het bewerken van verzamelingen en bits. De compiler verwerkt operatoren volgens bepaalde regels. Kennis van deze regels is onontbeerlijk om expressies op de juiste manier te kunnen gebruiken.

13.2 Rekenkundige operatoren

Onderstaande tabel bevat de rekenkundige operatoren en hun betekenis:

Operator:	Bewerking:	Bewerkt type:	Type resultaat:
+	Optellen	Integer,Real	Integer,Real
-	Aftrekken	Integer,Real	Integer,Real
*	Vermenigvuldigen	Integer,Real	Integer,Real
\	Delen	Integer,Real	Real
DIV	Delen	Integer	Integer
MOD	Restberekening	Integer	Integer

Met integer-type wordt niet het type Integer bedoeld, maar de typen die een geheel getal bevatten. Word, Byte en Longint zijn alle integer-typen. Gehele getallen staan hier tegenover gebroken getallen die worden gerepresenteerd door het type Real.

In de laatste kolom kun je zien dat het resultaat van de bewerking een ander type kan opleveren. Als twee integers een bewerking ondergaan, dan is het resultaat van deze bewerking ook een integer. Als één van operanden een real is, is het resultaat van de bewerking altijd een real.

Het programma EXPR_1 laat het resultaat van bewerkingen van verschillende expressies zien:

```
PROGRAM EXPR_1;
USES CRT;
VAR
   INT_GETAL  :Integer;
   REAL_GETAL :Real;

BEGIN
   ClrScr;
   REAL_GETAL := 24.675;
   INT_GETAL  := 20;
   Write('De bewerking van twee integers ');
   Write('resulteert in een geheel getal: ');
   Writeln(INT_GETAL DIV INT_GETAL);
   Write('De uitkomst van REAL_GETAL * INT_GETAL =');
   Writeln(REAL_GETAL * INT_GETAL);
   Write('Netjes geschreven is dat :');
   Writeln(REAL_GETAL * INT_GETAL:4:2);
   INT_GETAL := Round(REAL_GETAL);
   Write('Met hulp van Round staat in INT_GETAL:');
   Writeln(INT_GETAL);
   INT_GETAL := Trunc(REAL_GETAL);
   Write('Met hulp van Trunc staat in INT_GETAL:');
   Writeln(INT_GETAL);
   Readln
END.
```

Regels:Toelichting:

[1] 3-5 Declareer een integer- en een real-variabele.
[2] 8-9Geef beide variabelen een waarde.
[3] 10-12 Deel een integer door een integer en laat het resultaat op het scherm zien.
[4]13-14Vermenigvuldig een integer met een real en laat het resultaat op het scherm zien.
[5]15-16 Zet een real op het scherm met twee cijfers achter de decimale punt.
[6]17-19Gebruik de conversiefunctie Round om de afgeronde waarde van de real in een integer te zetten.
[7]20-22Gebruik de conversiefunctie Trunc om de waarde in REAL_GETAL vóór de decimale punt in de integer te zetten.

Toelichting:

[1]In EXPR_1.PAS worden een integer en een real gedeclareerd om de werking van de operatoren te demonstreren.

[2]In de variabele INT_GETAL, die van het type Integer is, zetten we een geheel getal. In de variabele REAL_GETAL zetten we een gebroken getal.

[3]Als twee operanden beide van het type Integer zijn, is het resultaat van de bewerking ook van het type Integer. Het delen van INT_GETAL door INT_GETAL heeft als uitkomst het gehele getal 1. Voor het delen van integers wordt de operator DIV gebruikt.

[4]Als in de bewerking één van de operanden van het type Real is, zal de uitkomst van de bewerking ook van het type Real zijn. De vermenigvuldiging van REAL_GETAL met INT_GETAL, levert een gebroken getal op in de wetenschappelijke notatie.

[5]Het getal in de wetenschappelijke notatie kan voor ons beter leesbaar worden door een formaat mee te geven. "REAL_GETAL * INT_GETAL:4:2" toont de uitkomst van de vermenigvuldiging op het scherm, waarbij vier posities gereserveerd zijn voor de cijfers vóór de decimale punt en twee posities voor de cijfers daarachter.

[6]De waarde in een integer kan aan een real gegeven worden. Omdat een integer een geheel getal is, staat er dan geen waarde na de decimale punt. Andersom kan de waarde in een real niet

zomaar aan een integer gegeven worden. Hiervoor moet van de real eerst een geheel getal gemaakt worden. De Turbo Pascal-functie Round krijgt als parameter een real mee en retourneert een afgerond geheel getal. Alles boven de 0.5 wordt naar boven afgerond en onder de 0.5 naar beneden.

[7]Naast Round beschikken we ook nog over de Turbo Pascal-functie Trunc. Deze functie retourneert het getal vóór de decimale punt als geheel getal. De Turbo Pascal-functie Int doet hetzelfde, maar deze functie retourneert een getal van het type Real.

13.3 Unary operatoren

De rekenkundige operatoren die in de vorige paragraaf behandeld zijn, worden ook wel binary operatoren genoemd. "Binary" wil zeggen dat er steeds twee operanden nodig zijn om met deze rekenkundige operatoren te kunnen werken. Naast binary operatoren beschikken we ook over rekenkundige operatoren die met één operand werken; die heten unary operatoren:

Operator:	Bewerking:	Bewerkt type:	Type resultaat:
+	Plusteken	Integer,Real	Integer,Real
-	Minteken	Integer,Real	Integer,Real

Een minteken vóór een getal duidt aan dat we te maken hebben met een negatief getal. Een plusteken representeert een positief getal. Een plusteken laten we meestal achterwege. Het resultaat van de bewerking met een plusteken of een minteken is steeds van hetzelfde type als de operand. Om een getal, ongeacht of het een positief of een negatief getal is, altijd positief weer te geven, beschikken we over de Turbo Pascal-functie Abs (van Absoluut). Abs krijgt als parameter een getalswaarde mee en retourneert een positief getal.

13.4 Logische operatoren

De logische operatoren die je kunt gebruiken zijn:

Operator:	Bewerking:	Bewerkt type:	Type resultaat:
NOT	Draait bits om	Integer	Integer
AND	Bitsgewijze AND	Integer	Integer
OR	Bitsgewijze OR	Integer	Integer
XOR	Bitsgewijze XOR	Integer	Integer
SHL	Schuift naar links	Integer	Integer
SHR	Schuift naar rechts	Integer	Integer

NOT-operator

In het rijtje van de tabel is de NOT-operator de enige unary operator. Dat wil zeggen dat NOT slechts met één operand werkt. De NOT-operator draait een bitpatroon om: alle nullen worden enen en alle enen worden nullen. Stel dat we een variabele GETAL van het type Byte hebben. We zetten in GETAL de waarde 0. In dat geval staan alle acht bits van GETAL op 0. De opdracht "NOT

GETAL" zet alle bits in GETAL op 1. Het resultaat van deze opdracht is dat GETAL de waarde 255 heeft.

AND-operator

De AND-operator voert een logische EN uit en werkt met twee operanden. Bij een logische EN worden twee bitpatronen vergeleken. In het resultaat worden de bits die in beide operanden op 1 staan, ook op 1 gezet. Uit het volgende voorbeeld:

Operand A : 10100111 = 167 decimaal
Operand B : 00000111 = 7 decimaal
AND : 00000111 = 7 decimaal

blijkt dat je de AND-operator kunt gebruiken om te zien of een bepaald bitpatroon in een veld voorkomt. In dit voorbeeld wordt het resultaat een 1 als zowel operand A als operand B op 1 staat. De uitkomst van "A AND B" levert volgens de ASCII-tabel het getal 7 op. De AND-operator werkt dus als een filter:

IF Operand_A AND Operand_B = 7 THEN Doe iets

OR-operator

De OR-operator zet het overeenkomstige bit op 1 als minstens in één van de operanden een bit op 1 staat:

Operand A : 00000111 = 7 decimaal
Operand B : 10100001 = 161 decimaal
OR : 10100111 = 168 decimaal

De OR-operator kan dus gebruikt worden om twee bitpatronen samen te voegen. Het resultaat is dat de bits die in minstens één van de operanden op 1 staan ook op 1 gezet worden. Staan de bits van beide operanden op 1, dan is het resultaat dus ook 1. Na een dergelijke operatie kan AND weer gebruikt worden om na te gaan of een bepaald bitpatroon in een veld voorkomt.

XOR-operator

Deze operator is een samentrekking van de woorden "eXclusive OR". Bij een XOR-operatie wordt een bit op 1 gezet als het overeenkomstige bit in de twee operanden een verschillende waarde hebben:

Operand A : 00000001 = 1 decimaal
Operand B : 00000001 = 1 decimaal
XOR : 00000000 = 0

Hier worden geen verschillende bits gevonden, met als gevolg dat de uitkomst 0 is. Als we deze uitkomst in operand A zetten en we voeren opnieuw een XOR uit dan krijgen we:

Operand A : 00000000 = 0
Operand B : 00000001 = 1 decimaal
XOR : 00000001 = 1 decimaal

Uit dit voorbeeld blijkt dat je XOR kunt gebruiken om iets uit te zetten als het aan staat en iets aan te zetten als het uit staat. Het functioneert dus als een schakelaar.

SHL- en SHR-operator

SHL en SHR zijn afkortingen van de woorden "Shift Left" en "Shift Right", ofwel schuif naar links en naar rechts. Ze worden ook wel schuif-operatoren genoemd:

Operand A : 00001111 = 15 decimaal
SHR 1 : 00000111 = 7 decimaal

We zien dat het bitpatroon één plaats naar rechts opgeschoven is. Aan de linkerkant wordt het bitpatroon aangevuld met zoveel nullen als nodig is. In dit geval is het patroon één plaats opgeschoven, omdat er "SHR 1" staat. Had er "SHR 2" gestaan, dan zouden dit twee plaatsen zijn geweest. Het naar rechts schuiven veroorzaakt zoals je ziet een deling door 2. In het volgende voorbeeld wordt naar links geschoven:

Operand A : 00001111 = 15 decimaal
SHL 1 : 00011110 = 30 decimaal

De SHL-operatie schuift hier de bits naar links en vult het bitpatroon aan de rechterkant aan met nullen. Het naar links schuiven veroorzaakt telkens een machtsverheffing met 2. Dit machtsverheffen lukt net zo lang als er bits voorradig zijn.

In het volgende voorbeeld is de uitkomst niet 480 ($15 * 2^5$), maar 224. Dit komt omdat er één bit over de rand geduwd is. Dit wordt "overflow" genoemd:

Operand A : 00001111 = 15 decimaal
SHL 5 : 11100000 = 224 decimaal

13.5 Boolean operatoren

De bewering "25 > 30" is een logische expressie. Een dergelijke expressie kan maar twee waarden hebben: "waar" of "niet waar". Hier wordt beweerd dat 25 groter is dan 30. Deze expressie is dus "niet waar". In Turbo Pascal zal de expressie de waarde False krijgen. Ook voor het evalueren van logische expressies beschikt Turbo Pascal over operatoren:

Operator:	Bewerking:	Bewerkt type:	Resultaat type:
NOT	Ontkenning	Boolean	Boolean
AND	Logische EN	Boolean	Boolean
OR	Logische OF	Boolean	Boolean
XOR	Logische XOR	Boolean	Boolean

NOT-operator

De NOT-operator keert de bewering om. "NOT X > Y" betekent dat de expressie "waar" wordt als X kleiner dan Y is. "NOT X = Y" maakt de bewering "waar" als X niet gelijk aan Y is.

AND-operator

De AND-operatie geeft de waarde True terug als de expressies aan weerszijden True zijn. Beide beweringen moeten dus "waar" zijn.

OR-operator

De OR-operatie geeft de waarde True terug als ten minste één van de beweringen "waar" is. Als geen van de expressies "waar" is, dan retourneert OR False.

XOR-operator

De XOR-operator geeft alleen de waarde True terug als één van de operanden True is en de andere False. In alle andere gevallen retourneert XOR de waarde False.

13.6 String-operator

Voor het werken met strings is er slechts één operator beschikbaar:

Operator:	Bewerking:	Bewerkt type:	Resultaat type:
+	Verbinding	String, Char	String

Deze operator verbindt twee strings of een string en een Char-variabele met elkaar:

Zin := Zin1 + Zin2 + Letter;

De Turbo Pascal-functie Concat doet precies hetzelfde:

Zin := Concat(Zin1,Zin2,Letter);

Het gebruik van de operator + is wat duidelijker, maar het resultaat is hetzelfde.

13.7 Set-operatoren

Ook sets kunnen in Turbo Pascal bewerkt worden met operatoren:

Operator:	Bewerking:	Bewerkt type:	Resultaat type:
+	Vereniging	Overeenkomstige sets	Set
-	Verwijdering	Overeenkomstige sets	Set
*	Behoud	Overeenkomstige sets	Set

Veronderstel dat we twee sets van het type Byte hebben:

Set_1 := [1,2,3,4];
Set_2 := [4,5,6];

De bewerking:

Set_1 := Set_1 + Set_2;

levert de elementen [1,2,3,4,5,6] op. De operator + heeft beide sets samengevoegd.

De bewerking:

Set_1 := Set_1 - Set_2;

heeft [1,2,3] als resultaat. Zowel in Set_1 als in Set_2 komt het cijfer 4 voor. De min-operator heeft dit cijfer verwijderd.

In plaats van de minus-operator kun je ook de procedure Exclude aanroepen. Aan deze procedure wordt als VAR-parameter een variabele van het type Set en een te verwijderen waarde doorgegeven.

Als we opgeven:

Set_1 := Set_1 * Set_2;

krijgen we het element [4] terug. Het maal-teken zorgt ervoor dat alleen overeenkomstige elementen van de set gehandhaafd blijven. Het cijfer 4 is de enige waarde die in beide sets voorkomt. In de nieuwe set is dit cijfer dan ook de enige overgebleven waarde.

13.8 Adres-operator

De operator om de adressen van variabelen, functies en procedures te berekenen, is het apestaartje @ (teken 64 uit de ASCII-tabel). Dit is een unary operator die het gezochte adres in de operand zet. De Turbo Pascal-functie Addr(variabele-naam) levert het adres van de doorgezonden variabele en doet feitelijk hetzelfde als het apestaartje.

13.9 Relationele operatoren

De laatste tabel bevat de operatoren die gebruikt kunnen worden om de relatie tussen operanden vast te leggen. Hierbij is het van belang dat je weet wat een eenvoudig type is. Een eenvoudig type definieert een geordend pakket aan waarden. In die zin zijn de typen Byte, Char, Word, enzovoort eenvoudige typen. Ook het type Real behoort hier toe. In de tabel wordt onder eenvoudig type steeds verstaan: typen die een waarde aan elkaar kunnen overdragen. De waarde in een variable van het type Byte kan niet zo maar overgedragen worden aan een variabele van het type Char:

Operator:	Bewerking:	Bewerkt type:	Resultaat:
=	Gelijk	Eenvoudige typen: Pointer, Set, String	Boolean
<>	Niet gelijk	Eenvoudige typen: Pointer, Set, String	Boolean
<	Kleiner dan	Eenvoudig type: String	Boolean
>	Groter dan	Eenvoudig type: String	Boolean
<=	Kleiner of gelijk	Eenvoudig type: String Sets	Boolean
>=	Subset van/ Groter of gelijk	Eenvoudig type: String Sets	Boolean
IN	Superset	Linker operand kan ieder opsommingstype zijn. De rechterkant is een set waarvan de inhoud bij de linker operand past.	Boolean

Eenvoudige typen

Als de operanden van de operatoren uit de tabel van een eenvoudig type zijn, dan moeten ze wel bij elkaar passen, maar ze mogen wel verschillend zijn. Dus als de ene operand van het type Integer is, mag de andere best van het type Real zijn.

Strings

Alle strings kunnen met elkaar vergeleken worden. Als uitgangspunt voor de vergelijking wordt de ordening van tekens in de ASCII-set genomen. De operatoren houden rekening met de lengte van de set. De volgende vergelijking:

'Aa' < 'a';

heeft als uitkomst de waarde True. De hoofdletter "A" heeft volgens de ASCII-tabel een lagere waarde dan de kleiner letter "a". Dat in de vergelijking de linker operand behalve de "A" ook nog een "a" bevat doet niet terzake.

Pointers

De operatoren = en < en > kunnen gebruikt worden voor pointers. De twee pointer-typen moeten dan wel bij elkaar passen.

Sets

De operator = geeft bij sets de uitkomst True als beide operanden precies dezelfde inhoud hebben. De operator <= is True als de inhoud van de linker operand ook voorkomt in de rechter operand. Het ligt dan voor de hand dat de operator >= kijkt of de inhoud van de rechter operand ook voorkomt in de linker operand.

13.10 Regels voor het werken met operatoren

Bij het maken van een expressie is het belangrijk om de regels goed toe te passen. Als je je niet aan de regels houdt, kan dat

resulteren in verkeerde uitkomsten.

Een expressie wordt van links naar rechts uitgevoerd. De tussen haakjes geplaatste delen van de expressie worden het eerst uitgevoerd. De operatoren in een expressie hebben een bepaalde prioriteit. Dat wil dus zeggen dat een expressie weliswaar van links naar rechts uitgewerkt wordt, maar dat eerst de operatoren met de hoogste prioriteit behandeld worden, vervolgens de operatoren met een lagere prioriteit, enzovoort. De volgende tabel bevat de prioriteiten van de verschillende operatoren:

Operatoren:	Prioriteit:	Categorie operator:
@ NOT	Eerste	Unary
* / DIV MOD AND	Tweede	Vermenigvuldiging
SHL SHR	Derde	Toevoeging
+ - OR XOR	Vierde	Relationeel
= <> < > <= >=	Laagste	Vergelijking

Laten we de waarde van de volgende expressie bepalen:

10 + 1 AND 5 * 10;

Volgens de tabel op de vorige pagina hebben de operatoren * en AND gelijke prioriteit. In zo'n geval wordt de meest linker operator in de expressie als eerste uitgevoerd. Als eerste wordt dus het deel "1 AND 5" uitgevoerd:

```
   0110001 = 1 decimaal
      0110101 = 5 decimaal
AND: 0110001 = 1 decimaal
```

De uitkomst van "1 AND 5" levert volgens de ASCII-tabel het decimale getal 1 op. Daarna heeft de vermenigvuldiging de hoogste prioriteit. Hier wordt de uitkomst van "1 AND 5" genomen om te vermenigvuldigen met 10. Dit is dus: 1 * 10 = 10. Deze laatste uitkomst wordt gebruikt als operand bij de plus-operator. De uitkomst van de expressie is dan:

10 + 10 = 20;

Zouden we haakjes in de expressie gebruiken, dan krijg je andere uitkomsten. Met de expressie:

10 + 1 AND (5 * 10);

wordt eerst het deel tussen haakjes uitgevoerd. De uitkomst hiervan is 50. Daarna wordt "1 AND 50" uitgerekend. De uitkomst hiervan is 0. Deze 0 wordt opgeteld bij het getal 10 die voor de plus-operator staat. De uitkomst is dan: 10 + 0 = 10.

13.11 Voorbeeldprogramma

Het wordt tijd om gewapend met deze kennis een programma te maken. In het volgende programma kan de gebruiker een geheel positief getal opgeven. Het programma geeft vervolgens de binaire notatie van dit getal.

Om een dergelijk programma te kunnen maken, bepalen we eerst uit welke onderdelen het zal bestaan:

- Behalve het hoofdprogramma hebben we een procedure nodig om een getal in te lezen vanaf het toetsenbord. Deze procedure moet ervoor zorgen dat uitsluitend een geheel en positief getal ingevoerd mag worden. Als er wel iets anders dan een geheel, positief getal ingevoerd wordt, dan moet er een foutmelding volgen en moet de gebruiker van het programma een nieuwe kans krijgen om een getal in te voeren.

- Als er een correct getal is ingevoerd, dan moet in een procedure of functie het ingevoerde getal omgezet worden naar binaire notatie. Deze binaire versie wordt vervolgens op het scherm getoond.

- Om een getal in te lezen en in binaire notatie weer te geven, hebben we een venster nodig. Het staat mooier als dit venster voorzien is van een frame. Bovendien moeten de voor- en achtergrondkleur instelbaar zijn. We moeten dus een procedure maken waaraan je kunt doorgeven hoe groot het venster moet zijn en in welke kleuren het moet verschijnen. De procedure moet zelf uitrekenen hoe het frame getekend moet worden.

- Omdat we een dergelijk scherm waarschijnlijk in meer programma’s nodig zullen hebben, maken we een unit DIVERSEN waarin de procedure Venster opgenomen is. Op deze manier hoeven we in toekomstige programma’s alleen maar de regel "USES DIVERSEN" op te nemen om de beschikking te hebben over het venster.

Eerst maken we de unit DIVERSEN:

UNIT DIVERSEN;

```
Interface
USES CRT;

CONST
   VX1: Byte = 10;
   VX2: Byte = 70;
   VY1: Byte =  5;
   VY2: Byte = 15;

PROCEDURE Venster(X1,Y1,X2,Y2,Achtergrond,Voorgrond:Byte);

Implementation

PROCEDURE Venster(X1,Y1,X2,Y2,Achtergrond,Voorgrond:Byte);
VAR
   I,J,X,Y,LX1,LY1,LX2,LY2: Byte;

BEGIN
   Window(X1,Y1,X2,Y2);
   TextBackground(Achtergrond);
   TextColor(Voorgrond);
   ClrScr;
   LX1 := 2;
   LY1 := 1;
   LX2 := X2 - X1;
   LY2 := Y2 - Y1 + 1;
   GotoXY(LX1,LY1);
   Write(#201);
   GotoXY(LX1,LY2);
   Write(#200);
   FOR I := 1 TO 2 DO
   BEGIN
      CASE I OF
         1: Y := LY1;
         2: Y := LY2;
      END;
      FOR J := LX1 + 1 TO LX2 - 1 DO
      BEGIN
         GotoXY(J,Y);
         Write(#205)
      END
   END;
   GotoXY(LX2,LY1);
   Write(#187);
   GotoXY(LX2,LY2);
```

```
   Write(#188);
   FOR I := 1 TO 2 DO
   BEGIN
      CASE I OF
         1: X := LX1;
         2: X := LX2;
      END;
      FOR J := LY1 + 1 TO LY2 - 1 DO
      BEGIN
         GotoXY(X,J);
         Write(#186)
      END
   END;
   Window(X1+2,Y1+1,X2-2,Y2-1)
END;

END.
```

Regels:Toelichting:

```
     [1]4-8Declareer voorgedefinieerde constanten voor de
          afmetingen van het venster.
   [2]9   Maak de procedure Venster algemeen toegankelijk.
11-56 Procedure Venster(X1,Y1,X2,Y2,Achtergrond,Voorgrond:Byte).
 [3]12-13Declareer de benodigde lokale variabelen.
   [4]15-18Roep de Turbo Pascal-procedure Window aan en geef als
          parameters de vermelde afmetingen door. Zet de
          achtergrond- en de voorgrondkleur op de vermelde
          kleurwaarden en veeg het scherm schoon.
   [5]19-22   Bereken de coördinaten voor het frame en plaats
          deze in LX1, LY1, LX2 en LY2.
    [6]23-26Plaats in de linker bovenhoek en in de linker
          benedenhoek het hoektekentje voor het frame.
   [7]27-38Ga een FOR-lus in die twee keer wordt uitgevoerd. Voer
          hier een CASE OF uit om te bepalen welke waarde Y moet
          krijgen. Voer dan in de FOR-lus opnieuw een FOR-lus
          uit, die zo vaak uitgevoerd wordt als nodig is om de
          horizontale lijnen van het frame te tekenen.
   [8]39-42Zet de hoektekentjes van de rechter bovenhoek en de
          rechter benedenhoek op het scherm.
   [9]43-54Ga een FOR-lus in met dezelfde structuur als de lus
          van regel 27, maar nu voor het tekenen van de verticale
          lijnen.
   [10]55 Roep opnieuw de Turbo Pascal-procedure Window aan om
          een venster binnen het frame te maken.
57        Einde van de unit.
```

Toelichting:

[1] In de eerste plaats wordt een viertal getypeerde constanten gedeclareerd. Deze kunnen gebruikt worden om een scherm van 60 tekens breed en 10 tekens hoog te maken. Uiteraard kan de programmeur deze waarden veranderen. Werken met de gedefinieerde waarden heeft als voordeel dat als je besluit alsnog een andere afmeting te kiezen, je hiertoe slechts op één plaats de waarden hoeft te veranderen. Pas waar mogelijk deze werkwijze toe. Het kan een hoop veranderen en aanpassen besparen.

[2]Door vermelding van de procedure Venster in het Interface-gedeelte van de unit, wordt de procedure Venster toegankelijk voor programma’s en units die in hun USES-regel de unit DIVERSEN opnemen.

[3]Voor het berekenen van de coördinaten van het frame en voor

index-doeleinden hebben we een aantal lokale variabelen nodig.

[4]De afmetingen van het venster worden aan de procedure Venster als parameter meegegeven, evenals de achtergrondkleur en de voorgrondkleur. Deze afmetingen worden doorgegeven aan de Turbo Pascal-procedure Window. Als de voor- en achtergrondkleur gezet zijn, wordt met ClrScr het zojuist gemaakte venster schoongeveegd en hebben we de beschikking over een venster in de gewenste afmeting en kleur.

[5]Omdat in het nieuwe venster voor de X-lijn de waarden 1 tot en met 60 en voor de Y-lijn de waarden 1 tot en met 10 geldig zijn, moeten er nieuwe coördinaten berekend worden om het frame te kunnen tekenen. Aan de linkerkant komt het frame 1 positie van de kant te staan. De lokale variabele LX1 komt dan op 2 te staan en LY1 komt op 1 te staan. De coördinaten van de linker bovenhoek zijn dan 2,1. LX2 moet de beginpositie voor de rechterkant van het venster bevatten. De aftreksom van de parameters X2 en X1 geeft de horizontale afmeting van het venster. De berekening "Y2-Y1+1" levert de coördinaat op voor de onderste lijn van het scherm.

[6]Voor het frame gebruiken we de grafische tekens uit de ASCII-tabel. Hier vinden we tekens voor horizontale en verticale lijnen, maar ook tekens om hoeken te verbinden. Voor de linker bovenhoek en de linker benedenhoek worden de bijpassende tekentjes op de juiste plaats gezet.

[7]Vervolgens moeten we aan de onderkant en aan de bovenkant van het scherm een horizontale lijn tekenen. De opdrachten voor deze twee lijnen zijn bijna identiek. Het enige verschil is de plaats waar de lijn getrokken moet worden: de één bovenaan, de andere onderaan. Deze plaats wordt bepaald door de lokale variabele Y. In de FOR-lus die twee keer uitgevoerd wordt, krijgt Y in de eerste doorloop de waarde van LY1. In de tweede doorloop krijgt Y de waarde van LY2. Als deze waarde bepaald is met behulp van het CASE OF-statement, gaan we een nieuwe FOR-lus in. Dit betekent dat er nu een FOR-lus in een FOR-lus uitgevoerd wordt. Dit wordt ook wel het nesten van lussen genoemd. Deze nieuwe FOR-lus wordt uitgevoerd van LX1+1 tot LX2-1. In deze variabelen staan de coördinaten voor de horizontale afmeting van het frame. Het optellen en aftrekken is nodig voor de ruimte van de hoektekens.

[8]Als de horizontale lijnen gemaakt zijn, zetten we de rechter hoektekens.

[9]Vervolgens wordt om de verticale lijnen van het frame te tekenen, een soortgelijk mechanisme gebruikt als bij de horizontale lijnen.

[10]Tenslotte wordt de Turbo Pascal-procedure Window aangeroepen om een venster te maken binnen de lijnen van het frame. Je zult je misschien afvragen waar dat nu weer voor nodig is. Als je op het scherm tekst hebt afgedrukt en je wilt het weer schoonvegen met ClrScr of ClrEol, zou ook het frame verdwijnen. Door een venster binnen het venster te maken, behouden we het frame.

Nu kunnen we aan het eigenlijke programma beginnen:

PROGRAM BITS;

```
USES CRT, DIVERSEN;

FUNCTION ToonGetal(Getal:Word):Char;
VAR
   BITPATROON: String[17];
   SJABLOON  : Word;
   I         : Byte;
   CH        : Char;
BEGIN
   SJABLOON := 1;
   BITPATROON := '';
   Venster(VX1,VY1,VX2,VY2,4,15);
   GotoXY(5,3);
   Write('De binaire voorstelling van ',GETAL,' is:');
   FOR I := 0 TO 15 DO
   BEGIN
      IF GETAL AND SJABLOON = 1 THEN CH := '1'
      ELSE
      CH := '0';
      BITPATROON := BITPATROON + CH;
      GETAL := GETAL SHR 1
   END;
   GotoXY(10,5);
   Insert(#32,BITPATROON,9);
   Write(BITPATROON);
   GotoXY(5,9);
   Write('Wilt U nog een getal zien? (J/N)');
   ToonGetal := Readkey
 END;

PROCEDURE VoerGetalIn;
VAR
   INVOER         :String[20];
   GETAL          :Longint;
   CODE           :Integer;
   OK, NIEUWGETAL :Boolean;
BEGIN
   NIEUWGETAL := True;
   WHILE NIEUWGETAL DO
   BEGIN
      Venster(VX1,VY1,VX2,VY2,1,14);
      OK := False;
      WHILE NOT OK DO
      BEGIN
         GotoXY(5,5);
```

```
         Write('Voer geheel positief getal in: ');
         GotoXY(5,40);
         ClrEol;
         Readln(INVOER);
         Val(INVOER,GETAL,CODE);
         OK := (CODE = 0) AND (INVOER[1] <> '-')
               AND (GETAL <= 65535);
         IF NOT OK THEN
         BEGIN
            GotoXY(5,8);
            IF GETAL > 65535 THEN
            BEGIN
               Write(#7,'Getal mag niet groter zijn');
               Write(' dan 65535.  Druk op Enter')
            END
               ELSE
            BEGIN
               Write(#7'Een geheel positief getal svp!');
               Write('  Druk op Enter')
            END;
            Readln;
            GotoXY(5,8);
            ClrEol
         END
      END;
      NIEUWGETAL := ToonGetal(GETAL) IN ['J','j']
   END
END;

BEGIN
   TextBackground(0);
   ClrScr;
   VoerGetalIn;
   TextBackGround(0);
   Window(1,1,80,25);
   ClrScr
END.
```

Regels:Toelichting:

[1]2 Gebruik de units CRT en DIVERSEN.

3-29Function ToonGetal(Getal:Word):Char.

[10]4-8 Declareer de benodigde lokale variabelen.

[11]10-11Zet SJABLOON op 1 en maak van BITPATROON een lege string.

12 Roep de procedure Venster (uit de unit DIVERSEN) aan.

[12]14-25Maak een bitpatroon van de waarde die in de parameter Getal staat en toon de uitkomst op het scherm.

[13]26-28Vraag of er nog een getal getoond moet worden en geef de functie ToonGetal de waarde van de ingedrukte toets.

30-72Procedure VoerGetalIn.

[4]31-35Declareer de benodigde lokale variabelen.

[5]37 Zet de lokale variabele NIEUWGETAL op True.

[6]38-67Ga een WHILE-lus in die duurt tot het programma beëindigd moet worden.

[7]40 Roep de procedure Venster (uit de unit DIVERSEN) aan.

[8]41Zet de lokale variabele OK op False.

[9]42-69Ga een WHILE-lus in waarin een getal ingelezen wordt en tevens de bruikbaarheid van de invoer gecontroleerd wordt. Verlaat de lus als de invoer correct is.

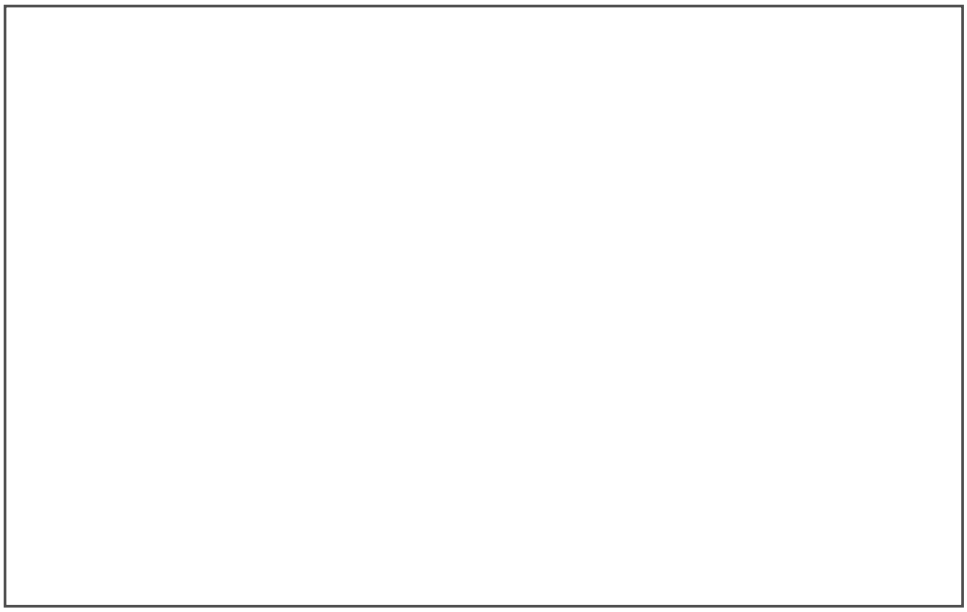
[10]70 Roep de functie ToonGetal aan en geef als parameter de variabele GETAL mee. Zet NIEUWGETAL op True als de geretourneerde waarde van ToonGetal een "J" of een "j" is.

73-80Hoofdprogramma.

[2]74-75Zet de achtergrondkleur op zwart en veeg het scherm schoon.

[3]76 Ga naar de procedure VoerGetalIn.

[14]77-79Maak het scherm weer zwart en geef het z’n oorspronkelijke afmetingen. Veeg ten slotte het scherm schoon.



Afbeelding 13

Toelichting:

[1]In de USES-regel wordt nu naast CRT ook de unit DIVERSEN vermeld. Daarom kan BITS.PAS de globale procedure Venster uit de unit DIVERSEN gebruiken.

[2]Omdat de vensters waarin een getal wordt ingelezen en waarin de uitkomst wordt getoond, moeten afsteken tegen de achtergrond, wordt eerst de achtergrondkleur op zwart gezet en wordt ClrScr gebruikt om het scherm schoon te vegen.

[3]Onmiddellijk daarna wordt naar de procedure VoerGetalIn gesprongen.

[4]We declareren hier onder andere de lokale variabele INVOER. Deze variabele is van het type String, terwijl we een getal moeten inlezen. Er wordt hier een string gebruikt, omdat daar alle mogelijke tekens in mogen staan. Als we hier een numerieke variabele zouden gebruiken, zou het programma telkens onderbroken worden als er andere invoer dan een getal zou worden ingegeven. Nu wordt de invoer eerst onderzocht. Als het een geldig getal is, wordt het omgezet naar een waarde voor de variabele GETAL, die op haar beurt weer naar de functie ToonGetal gestuurd wordt.

[5]De lokale variabele NIEUWGETAL maken we True. Dit is nodig omdat we anders niet de WHILE-lus op de volgende opdrachtregel in zouden gaan.

[6]De WHILE-lus wordt uitgevoerd tot de functie ToonGetal een andere waarde retourneert dan een "J" of een "j".

[7]Nu we in de lus zijn, wordt eerst de procedure Venster aangeroepen om een venster te maken waarin de gebruiker gevraagd zal worden een getal in te voeren.

[8]OK wordt False, zodat we de WHILE-lus op de volgende regel in kunnen gaan.

[9]Deze geneste WHILE-lus wordt gebruikt om een waarde in de string INVOER in te lezen. Als INVOER een waarde bevat, wordt deze met behulp van de Turbo Pascal-procedure Val omgezet naar een numerieke waarde voor GETAL.

OK krijgt nu de waarde True als CODE op 0 staat. Als dit het geval is, dan is de conversie van INVOER naar GETAL gelukt. Heeft CODE een andere waarde dan 0, dan is de conversie mislukt. CODE bevat dan de plaats in de string waar een onjuiste invoer (een ander teken dan een cijfer) staat. Omdat we uitsluitend gehele, positieve getallen accepteren die in de twee bytes van het type Word passen, mag INVOER[1] niet een min-teken bevatten, want dan hebben we te maken met een negatief getal. GETAL mag op haar beurt niet groter zijn dan 65535. Dit is de maximale waarde die in twee bytes past. Samenvattend: OK wordt True als CODE een 0 bevat én INVOER[1] geen minteken is én GETAL niet groter is dan 65535.

Als OK False is, wordt er nagegaan of GETAL groter is dan 65535 en wordt er op het beeldscherm een boodschap getoond. Anders wordt de gebruiker gevraagd een geheel en positief getal in te voeren.

In de Write-opdracht is "#7" opgenomen. Dit teken 7 uit de ASCII-tabel laat kort een pieptoontje horen.

Nadat de gebruiker op Enter gedrukt heeft, wordt met behulp van ClrEol de foutmelding uit het venster verwijderd, en wordt teruggekeerd naar het begin van de geneste WHILE-lus om een nieuwe invoer mogelijk te maken. ClrEol verwijdert een tekstregel van het scherm vanaf de positie van de cursor tot het einde van de regel.

[10]Als OK True is, wordt ToonGetal aangeroepen en wordt GETAL als parameter meegegeven. De lokale variabelen die in ToonGetal gedeclareerd worden, bestaan uit de string BITPATROON en de variabele SJABLOON. Deze variabele, die van het type Word is, wordt gebruikt om als operand te dienen bij het uitfilteren van de bits. De variabele I doet dienst als index, en CH wordt

gebruikt om elementen aan BITPATROON toe te voegen.

[11]Eerst wordt de variabele SJABLOON op 1 gezet. Dit betekent dat bit 0, de meest rechtse bit, op 1 staat en de overige bits op 0. De string BITPATROON wordt als een lege string gedeclareerd.

[12]We gaan nu een FOR-lus in die 16 keer uitgevoerd wordt. In de lus wordt een AND-operatie uitgevoerd met GETAL en SJABLOON als operanden. Als in beide operanden bit nul (het eerste bit) op 1 staat, dan is het resultaat van de AND-operatie 1 en krijgt de lokale variabele CH de waarde 1, anders krijgt CH de waarde 0. CH wordt met de opdracht: BITPATROON := BITPATROON + CH in BITPATROON gezet. Hierdoor wordt aan de linkerkant van de string een letter toegevoegd. Na deze bewerking worden de bits één plaats naar rechts geschoven. Bit 1 wordt nu bit 0. Opnieuw wordt de AND-operatie uitgevoerd. Dit gaat zo door tot alle zestien bits bekeken zijn. We hebben dan een string met zestien enen en/of nullen. Met behulp van de Turbo Pascal-procedure Insert wordt op de negende positie in de string een spatie ingevoegd. Hierdoor worden de twee bytes visueel van elkaar gescheiden. Vervolgens wordt de inhoud van BITPATROON op het scherm getoond.

[13]Ten slotte wordt er gevraagd of de gebruiker nog een getal wil zien. Met ReadKey krijgt de functie ToonGetal de waarde van de ingedrukte toets. Teruggekeerd in de procedure VoerGetalIn, wordt de lokale variabele NIEUWGETAL True als de procedure ToonGetal een "J" of een "j" retourneert. Als NIEUWGETAL True is, dan krijgt de gebruiker opnieuw de gelegenheid een getal in te voeren, anders wordt de procedure ToonGetal verlaten.

[14]We zijn nu weer terug in het hoofdprogramma. Hier wordt het scherm weer op een zwarte achtergrondkleur gezet. Met behulp van Window wordt er een venster gemaakt met de oorspronkelijke afmeting van het scherm. ClrScr veegt tenslotte het scherm schoon.

13.12 Opgaven

13.1 Breng in de volgende expressies zodanig haakjes aan, dat de gegeven uitkomsten kloppen:

a: **10 + 20 DIV 3 = 10**
b: **15 * 10 - 2 DIV 4 = 37**
c: **(NOT 25 < 50) AND (50 > 25) = False**
d: **8 SHL 2 * 2 DIV 3 + 2 = 25**
e: **20 + 2 * 6 - 3 SHL 2 = 44**

De evaluatie-faciliteit van de ontwikkelomgeving is heel handig om de uitkomsten te testen en om te zien welke invloed de plaats van de haakjes op de uitkomsten heeft. Deze faciliteit kun je vanuit het menu selecteren door achtereenvolgens te kiezen voor de opties «*Debug*» en «*Evaluate/modify*», of rechtstreeks met de toetscombinatie «*Ctrl-F4*».

225

225

225

247

247

247