6. Procedures en functies

6.1 Definities

Een procedure is een afgerond programmafragment met een naam. Je kunt naar zo'n stukje programma springen, waarna de uitvoering van de procedure begint. De sprong wordt uitgevoerd door in het hoofdprogramma, of in een andere procedure of functie eenvoudigweg de naam van de aan te roepen procedure of functie te noemen. Na uitvoering wordt teruggekeerd naar de eerstvolgende regel na de aanroep.

Een procedure is als volgt opgebouwd: het begint met het beschermde woord PROCEDURE, gevolgd door de naam en een puntkomma. Een procedure start bij het woord BEGIN en eindigt bij END;. Hiertussen staan de opdrachten die bij de procedure horen en uitgevoerd moeten worden. Stel dat we een procedure Tel_Op maken die de inhoud van de variabelen SOM en GETAL bij elkaar op moet tellen en de uitkomst in de variabele SOM moet zetten. De constructie kan dan de volgende zijn:

```
PROCEDURE Tel_Op;
BEGIN
   SOM := SOM + GETAL
END;
```

In het programma kunnen we iedere keer dat GETAL bij SOM moet worden opgeteld, volstaan met de opdracht "Tel_Op;". Hiermee wordt naar de procedure Tel_Op gesprongen en wordt GETAL bij SOM opgeteld.

Een functie doet in principe hetzelfde als een procedure. Er is echter wel een verschil. Een functie heeft altijd een type-aanduiding, net als een variabele. In een functie kan een berekening uitgevoerd worden, waarna de functie de waarde van de uitkomst van die berekening krijgt. Kort gezegd: een procedure voert alleen een bepaalde taak uit en een functie geeft bovendien een waarde terug. Het declareren van een functie wordt voorafgegaan door het beschermde woord FUNCTION, gevolgd door de functienaam, een dubbele punt, een type-aanduiding en ter afsluiting een puntkomma. Als we van de procedure Tel_Op een functie Optelling maken, dan krijgt deze de volgende vorm:

```
FUNCTION Optelling:Integer;
BEGIN
 Optelling := SOM + GETAL
END;
```

De functie Optelling kan nu als volgt aangeroepen worden:

```
SOM := Optelling;
```

De functie krijgt hiermee de waarde van de optelling van de variabelen SOM en GETAL. Deze waarde wordt in de variabele SOM gezet. Je ziet hier dus dat je een functie op dezelfde manier kunt gebruiken als een variabele.

Geef je procedures altijd een logische beschrijvende naam. “Tel_Op” is een goede keuze voor een procedure die een optelling doet. “Bier” of “dshask” zouden slechte keuzen zijn. Stel jouw vriend geeft jou een programma en je ziet ergens staan:

```
JHSKEW := dshask;
```

Zou je dan nog begrijpen waar dit stukje programmacode over gaat?

In het programma PROC_1 wordt een voorbeeld gegeven van het verschil tussen procedures en functies:

```
PROGRAM PROC_1;
USES CRT;
VAR
   ZIN: String;

   PROCEDURE Schrijf_Zin;
   BEGIN
      GotoXY(15,10);
      ZIN := 'Ik sta in de variabele ZIN';
      Write(ZIN)
   END;

   FUNCTION Functie_Zin:String;
   BEGIN
      Functie_Zin :=
      'Ik ben een waarde van de functie Functie_Zin'
   END;

BEGIN
   ClrScr;
   Schrijf_Zin;
   GotoXY(15,11);
   Write(Functie_Zin);
   Readln
END.
```

Regels: Toelichting:

3-4 Declareer een variabele ZIN van het type String.
[2] 5-10 Procedure Schrijf_Zin.

7Zet de cursor op de vijftiende positie van de tiende regel.
8Geef de variabele ZIN een waarde.
9Schrijf de inhoud van ZIN op de positie van de cursor.
[3] 11-15Function Functie_Zin.
13-14 Geef de functie Functie_Zin een waarde.
[1] 16-22Hoofdprogramma.
17 Veeg het scherm schoon.
[2] 18 Roep de procedure Schrijf_Zin aan.
19Ga naar de vijftiende positie van de elfde regel.
[3] 20Schrijf de waarde van de functie Functie_Zin op de positie van de cursor.
21Wacht op een druk op de Enter-toets.

Toelichting:

[1]Zoals je ziet, staat het hoofdprogramma onderaan in de programmacode. Eigenlijk is dit wel logisch. De compiler vertaalt de broncode regel voor regel naar de doelcode. De benamingen van functies en procedures worden vervangen door adressen. Als in het hoofdprogramma gesprongen wordt naar de procedure Schrijf_zin, moet die procedure al gecompileerd zijn, anders is er geen adres om naar toe te springen.

[2]Met het aanroepen van de procedure Schrijf_Zin, krijgt de variabele ZIN een waarde die wordt afgedrukt op het scherm. Na de uitvoering van deze taak wordt teruggekeerd naar het hoofdprogramma, in dit geval naar regel 19.

[3]Hier kunnen we duidelijk zien dat het de functie is die een waarde krijgt. Het gaat hier om een functie van het type String. De procedure Write zet de inhoud van de functie Functie_Zin op het scherm.

We zouden ook de variabele ZIN als volgt een nieuwe waarde kunnen geven:

```
ZIN := Functie_Zin;
Write(ZIN);
```

Dit zou tot hetzelfde resultaat leiden.

6.2 Lokale variabelen

Tot nu toe hebben we de variabelen gedeclareerd als zogenaamde globale variabelen. Dit wil zeggen dat deze variabelen overal in het programma gebruikt kunnen worden. Naast globale variabelen kent Pascal ook lokale variabelen. Lokale variabelen worden gedeclareerd in een functie of procedure en kunnen alleen in die functie of procedure gebruikt worden. Het volgende programma demonstreert het gebruik van lokale variabelen:

```
PROGRAM PROC_2;
USES CRT;

   PROCEDURE Schrijf_Zin;
   CONST
      X = 15;

   VAR
      ZIN: String;

   BEGIN
      GotoXY(X,10);
      ZIN:=’Ik sta in de lokale variabele ZIN’;
      Write(ZIN)
   END;

BEGIN
   ClrScr;
   Schrijf_Zin;
   Readln
END.
```

Het programma PROC_2 vertoont grote gelijkenis met PROC_1. De functie is weggelaten en de variabele ZIN is nu als lokale variabele van de procedure Schrijf_Zin gedeclareerd. Het gevolg van deze manier van declareren is dat de variabele ZIN nu uitsluitend in deze procedure bekend is. Zou er naast deze variabele ook nog een gelijknamige globale variabele ZIN gedeclareerd worden, dan zou de procedure Schrijf_Zin deze globale variabele ZIN niet herkennen.

6.3 Parameters

De waarden waarmee een procedure of een functie moet werken, kunnen aan de procedure of functie worden doorgegeven door middel van een zogenaamde parameterlijst. Als een procedure of functie van een parameterlijst voorzien is, dan wordt deze tussen haakjes achter de naam geplaatst. Als de procedure Schrijf_Zin een opgegeven zin zou moeten afdrukken op een opgegeven regel, dan wordt de declaratie:

PROCEDURE Schrijf_Zin(Regel:Byte;Zin:String);

Tussen de haakjes worden de naam en het type van de parameter opgegeven. De verschillende parameters worden door een puntkomma van elkaar gescheiden.

Een beginnend programmeur zal zich afvragen wat het nut is van het werken met parameters. Misschien wordt het duidelijk als we de zaak eens omdraaien. Als je zonder parameters werkt dan moeten procedures en functies met globale variabelen werken om hun taak te kunnen uitvoeren. Je krijgt dan een enorme lijst met globale variabelen. Als je wel met parameters werkt, kun je in een procedure of functie lokale variabelen declareren. Als je de diensten van een andere procedure of functie nodig hebt, dan stuur je de waarden die bewerkt moeten worden als parameter naar deze procedure of functie. Het gevolg is dat je de lijst met globale variabelen enorm kan beperken, wat typwerk bespaart en de overzichtelijkheid van het programma bevordert.

Het programma PROC_3 plaatst met behulp van de procedure Schrijf_Zin 25 zinnen op het scherm. Aan deze procedure worden het regelnummer en de inhoud van de zin doorgegeven:

```
PROGRAM PROC_3;
USES CRT;

   PROCEDURE Schrijf_Zin(Regel:Byte;Zin:String);
   CONST
      X = 15;

   BEGIN
      GotoXY(X,Regel);
      Write(ZIN)
   END;


   PROCEDURE Maak_25_Zinnen;
   VAR
      ZIN: String;
      I: Byte;

   BEGIN
      FOR I:= 1 TO 25 DO
      BEGIN
         Str(I,ZIN);
         ZIN := 'Dit is zin '+ ZIN;
         Schrijf_Zin(I,ZIN)
      END
   END;

BEGIN
   ClrScr;
   Maak_25_Zinnen;
   Readln
END.
```

Regels:Toelichting:

[6] 3-9Procedure Schrijf_Zin.

4-5 Declareer een constante X.

7Zet de cursor op de positie waarvan de waarde in X staat en op de regel die doorgegeven is via de parameter Regel.

8Schrijf de waarde in de parameter Zin op het scherm.

10-21Procedure Maak_25_Zinnen.

11-13Declareer een lokale variabele ZIN van het type String en een lokale variabele I van het type Byte.
[2] 15-20 Ga een lus in die 25 keer doorlopen wordt.
[3] 16 Converteer de waarde van I naar een string en zet die waarde in ZIN.
[4] 18 Voeg een waarde aan de inhoud van ZIN toe.
[5] 19Roep de procedure Schrijf_Zin aan en geef de waarden in I en ZIN door.
22-26Hoofdprogramma.

23 Veeg het scherm schoon.
[1] 24Roep de procedure Maak_25_Zinnen aan.
25Wacht op een druk op de Enter-toets.

Toelichting:

[1]Vanuit het hoofdprogramma wordt de procedure Maak_25_Zinnen aangeroepen.

[2]In deze procedure wordt een FOR-lus uitgevoerd die 25 keer wordt uitgevoerd. Na iedere keer dat de FOR-lus is doorlopen, wordt de variabele I met 1 verhoogd. Een FOR-lus wordt altijd een bepaald aantal keren uitgevoerd. In dit geval dus 25 keer. Een FOR-lus kan ook omgekeerd worden uitgevoerd. In plaats van:

```
FOR I := 1 TO 25 DO
```

declareer je dan:

```
FOR I := 25 DOWNTO 1 DO
```

[3]De conversie-procedure Str maakt van de waarde die in I staat een string en plaatst die waarde in de variabele ZIN. De eerste keer dat de lus doorlopen wordt, heeft I de waarde 1, dus staat er in ZIN: "1".

[4]Nu wordt met de opdracht:
ZIN := ’Dit is zin ’+ ZIN;

de tekst en de inhoud van ZIN samengevoegd. Dan staat er dus "Dit is zin 1".
[5]Vervolgens wordt de waarde van I en de inhoud van ZIN doorgegeven aan de procedure Schrijf_Zin. Dit gaat door tot de FOR-lus 25 keer doorlopen is.
[6]De procedure Schrijf_Zin ontvangt de waarde van I in de parameter Regel en de waarde van ZIN in de parameter Zin. De constante X is op 15 gezet. De cursor wordt geplaatst op de positie die aangegeven

wordt door X, op de regel die aangegeven wordt door de parameter Regel. De parameter Zin wordt afgedrukt.

6.4 VAR-parameters

Parameters kunnen met of zonder de toevoeging VAR gedeclareerd worden. Als een parameter zonder de toevoeging VAR gedeclareerd wordt, kunnen er in de procedure of functie allerlei veranderingen aangebracht worden, zonder dat dit consequenties heeft voor de variabele die als parameter wordt doorgezonden. Deze variabele blijft ongewijzigd. Gaat het evenwel om een VAR-parameter, dan wordt bij een wijziging in de doorgezonden waarde ook de variabele gewijzigd waar deze waarde in staat. Als je in een procedure of functie een VAR-parameter declareert, dan heeft deze parameter hetzelfde adres in het geheugen als de variabele die doorgezonden werd. Veranderingen die deze variabele ondergaat, worden weggeschreven op dit adres. Het gevolg hiervan is, dat de doorgezonden variabele ook veranderd. Deze heeft immers hetzelfde adres. Als een parameter zonder de toevoeging VAR gedeclareerd wordt, dan wordt er een nieuwe variabele aangemaakt waarin de waarde komt te staan die doorgestuurd wordt. In het Engels worden de twee methoden "calling by reference" en "calling by value" genoemd. Bij calling by reference betreft het een VAR-parameter. Een voorbeeld maakt het wel duidelijk:

```
PROGRAM PROC_4;
USES CRT;
   PROCEDURE Verander(VAR A:Integer;B:Integer);
   BEGIN
      Writeln('In Verander ontvangen: A = ',A,' B =',B);
      Writeln('Bij A en B wordt tien opgeteld');
      Inc(A,10);
      Inc(B,10);
      Writeln('In Verander staat nu: A = ',A,' B = ',B)
   END;

   PROCEDURE Stuur_door;
   VAR
      X, Y: Integer;
   BEGIN
      X := 10;
      Y := 100;
      Writeln('In Stuur_door staat in: X = ',X,
              '  Y = ',Y);
      Verander(X,Y);
      Writeln('Terug in Stuur_door staat nu in:
               X =', X, '  Y = ',Y)
   END;

BEGIN
   ClrScr;
   Stuur_door;
   Readln
END.
```

Regels:Toelichting:

[2] 3-10Procedure Verander.
5 Laat zien wat de ontvangen waarden in de parameters A en B zijn.
6-8 Tel bij beide parameters 10 op.
9 Laat zien wat de waarden in de parameters nu zijn.
[1] 11-22Procedure Stuur_door.
12-13 Declareer twee lokale variabelen van het type Integer.
15-16 Geef X en Y een waarde.
17-18 Laat zien welke waarden in X en Y staan.
19Roep de procedure Verander aan en geef de waarde in X en Y door.
[3] 20 Toon bij terugkomst in de procedure Stuur_door opnieuw de waarden in X en Y.
23-27Hoofdprogramma.
24 Veeg het scherm schoon.
[1] 25 Roep de procedure Stuur_door aan.
26 Wacht op het indrukken van de Enter-toets.

Toelichting:

[1] Als de procedure Stuur_door aangeroepen wordt, worden in deze procedure twee lokale variabelen gedeclareerd van het type Integer. Deze variabelen, X en Y, krijgen respectievelijk de waarden 10 en 100. Nadat op het scherm getoond is dat deze waarden werkelijk in de variabelen staan, wordt in de procedure Stuur_door de procedure Verander aangeroepen.

[2] In de procedure Verander wordt de waarde die in de lokale variabele X van de procedure Stuur_door stond, ontvangen in de VAR-parameter A. De variabele Y wordt ontvangen in de gewone parameter B. Op het scherm wordt getoond dat deze parameters inderdaad de waarden 10 en 100 bevatten. Daarna wordt er bij iedere parameter 10 opgeteld. A is nu 20 en B is 110. Ook deze waarden worden getoond.

[3] Teruggekeerd uit de procedure Verander in de procedure Stuur_door, worden opnieuw de waarden van X en Y getoond. Nu blijkt dat de waarde van X, die doorgegeven werd aan de VAR-parameter, met 10 verhoogd is en dat de waarde van Y, die doorgegeven werd aan de gewone parameter, nog steeds 100 is.

6.5 Overroeping

Onze procedure Schrijf_Zin in programma PROC_3 zette een zin neer op een bepaalde regel. Stel dat we daar spijt van krijgen en niet alleen de regel, maar ook de kolom willen meegeven. We veranderen daartoe de procedure als volgt:

```
PROCEDURE Schrijf_Zin(Kolom,Regel:Byte;Zin:String);

BEGIN
   GotoXY(Kolom,Regel);
   Write(ZIN)
END;
```

Nadat we dit veranderd hebben, valt op dat het programma niet meer compileert, immers, in de procedure Maak_25_Zinnen staat nog steeds:

Schrijf_Zin(I,ZIN)

Eén om dit op te lossen is dat we dit gewoon veranderen in:

Schrijf_Zin(15,I,ZIN)

Maar stel dat we een programma hebben met 10000 regels, waar onze procedure Schrijf_Zin een paar honderd keer gebruikt wordt en weten niet waar. Een paar honderd keer een aanpassing doen is al behoorlijk vervelend. Nog vervelender zou het worden als we in een boek hadden beschreven hoe Schrijf_Zin werkt en dat boek nu in de winkel zou liggen. Het veranderen van de procedure zou onmiddellijk tot gevolg hebben dat dat boek foute informatie zou bevatten.

Voor dit soort situaties bestaat er in Free Pascal een mechanisme genaamd procedureoverroeping. De Engelse term die je hiervoor vaak zult tegenkomen is “procedure overloading”. Procedureoverroeping betekent dat je twee procedures mag schrijven met dezelfde naam, maar met andere parameters. De compiler zoekt uit welke aangeroepen moet worden.

Zouden we dit in ons programma toepassen, dan zou het er als volgt uit komen te zien:

```
PROGRAM PROC_5;
USES CRT;

   PROCEDURE Schrijf_Zin(Kolom,Regel:Byte;Zin:String);

   BEGIN
      GotoXY(Kolom,Regel);
      Write(ZIN)
   END;

   PROCEDURE Schrijf_Zin(Regel:Byte;Zin:String);
      CONST
      X = 15;

   BEGIN
      Schrijf_Zin(X,Regel,Zin);
   END;

   PROCEDURE Maak_25_Zinnen;
   VAR
      ZIN: String;
      I: Byte;

   BEGIN
      FOR I:= 1 TO 25 DO
      BEGIN
         Str(I,ZIN);
         ZIN := 'Dit is zin '+ ZIN;
         Schrijf_Zin(I,ZIN)
      END
   END;

BEGIN
   ClrScr;
   Maak_25_Zinnen;
   Readln
END.
```

4-9Procedure Schrijf_Zin.

4 In plaats van alleen een regel, staat er nu ook een kolom in de parameterlijst.

7 In plaats van de constante X wordt nu de kolom in GotoXY gebruikt.

11-15 Dit is een procedure met dezelfde parameterlijst als voorheen. De procedure roept gewoon de nieuwe Schrijf_Zin aan, met X=15 als kolomnummer.

Het voordeel van deze oplossing is dat de procedure nog steeds gewoon op de oude manier aangeroepen kan worden, met alleen een regelnummer dus. We kunnen de rest van ons programma daarom zo laten als het is, en eventuele boeken die in de winkel liggen over Schrijft_Zin verstrekken nog steeds juiste informatie.

6.6 Het nesten van procedures en functies

In een procedure of functie kunnen weer andere procedures en functies opgenomen worden. Het opnemen van een procedure of functie in een andere procedure of functie noemt men "nesten". Een dergelijke procedure of functie is alleen toegankelijk in de procedure of functie waar deze is gedeclareerd. Het eerder besproken programma PROC_3 leent zich er uitstekend voor om herschreven te worden met een geneste procedure:

```
PROGRAM PROC_6;
USES CRT;
   PROCEDURE Maak_25_Zinnen;
   VAR
      ZIN: String;
      I: Byte;
         PROCEDURE Schrijf_Zin;
         CONST
            X = 15;
         BEGIN
            GotoXY(X,I);
            Write(ZIN)
         END;
   BEGIN
      FOR I := 1 TO 25 DO
      BEGIN
         Str(I,Zin);
         ZIN := 'Dit is zin '+ ZIN;
         Schrijf_Zin
      END
   END;
BEGIN
   ClrScr;
   Maak_25_Zinnen;
   Readln
END.
```

Toelichting:

De procedure Schrijf_Zin is nu een lokale procedure van de procedure Maak_25_Zinnen geworden. Het is ook niet meer nodig om met een parameterlijst te werken. De lokale procedure Schrijf_Zin kan werken met de lokale variabelen van de procedure Maak_25_Zinnen.

6.7 De organisatie in het geheugen

Voor het bewaren van globale variabelen en constanten is een afgezonderd deel van het geheugen gereserveerd. Dat deel heet het datasegment. Variabelen en constanten die in dit segment geplaatst worden, blijven daar gedurende de hele uitvoering van het programma staan. Bij lokale variabelen en parameters werkt dit anders. Daar wordt gebruik gemaakt van een afgezonderd deel van het geheugen dat we de stapel noemen. De Engelse term "stack" wordt hiervoor ook gebruikt. De stapel werkt volgens het principe dat wat er het eerst op gezet wordt, er als laatste weer afgehaald wordt.

Op de stapel worden de terugkeeradressen, de lokale variabelen en de parameters geplaatst. Als een procedure of functie wordt aangeroepen, wordt eerst het terugkeeradres op de stapel gezet. Pas dan wordt de sprongopdracht uitgevoerd. Het terugkeeradres is het adres van de eerste opdracht die volgt op de sprongopdracht waarmee de functie of procedure werd aangeroepen. De aangeroepen procedure zet ook zijn lokale variabelen en parameters op de stapel. Als van daaruit weer een procedure of functie wordt aangeroepen, wordt weer eerst het terugkeeradres op de stapel gezet.

Als een procedure of functie doorlopen is, wordt de geheugenruimte die op de stapel door de lokale variabelen en parameters wordt ingenomen, weer vrijgegeven. Het bovenste adres van de stapel is dan het terugkeeradres van waaruit de procedure of functie werd aangeroepen. Naar dit adres wordt teruggekeerd en het geheugen dat door het terugkeeradres werd ingenomen, wordt vrijgegeven. Lokale variabelen blijven dus net zolang in het geheugen staan als nodig is om de procedure of functie uit te voeren.

Als in een programma vaak van procedure naar procedure gesprongen wordt, en er zijn in deze procedures grote datastructuren gedeclareerd, dan is er vanzelfsprekend een grote stapel nodig.

Je kunt de grootte van de stapel instellen door op de menubalk achtereenvolgens voor «*Options*» en «*Memory sizes*» te kiezen. Standaard staat de stapel ingesteld op 16384 bytes. De maximale grootte van de stapel is 65536 bytes.

6.8 Recursie

Een vreemd verschijnsel doet zich voor als een procedure of functie zichzelf aanroept. We noemen dat recursie. In de vorige paragraaf hebben we geleerd dat bij het aanroepen van een procedure of functie een terugkeeradres op de stapel gezet wordt, met de lokale variabelen en parameters van de aangeroepen procedure of functie. Als een procedure zichzelf aanroept, gebeurt hetzelfde. Het terugkeeradres wordt op de stapel gezet en vervolgens wordt opnieuw begonnen met de uitvoering van de procedure door het op de stapel zetten van de lokale variabelen. Zo ontstaat er een stapel van steeds dezelfde terugkeeradressen met lokale variabelen die telkens een andere waarde hebben.

Het volgende programma geeft hiervan een voorbeeld:

```
PROGRAM PROC_7;
USES CRT;

PROCEDURE Opnieuw(I:Byte);
BEGIN
   Inc(I,1);
   Writeln(I);
   IF I < 10 THEN Opnieuw(I);
   Writeln(I)
END;

BEGIN
   ClrScr;
   Opnieuw(0);
   Readln
END.
```

Regels:Toelichting:

[2]3-9 Procedure Opnieuw.
[3]5 Verhoog de parameter I met 1.
[4]6Zet de waarde van I op het scherm.
[5] 7 Als I kleiner is dan 10, roep dan de procedure Opnieuw aan en geef I als parameter mee.
8Zet de waarde van I op het scherm.

10-14Hoofdprogramma.

11Veeg het scherm schoon.

[1]12Roep de procedure Opnieuw aan en geef als parameter een 0 mee.

13 Wacht op het indrukken van de Enter-toets

Toelichting:

[1]Als vanuit het hoofdprogramma de procedure Opnieuw wordt aangeroepen, wordt als parameter een 0 meegegeven. Het terugkeeradres dat op de stapel gezet wordt, is het adres van regel 12.
[2]Aangekomen in de procedure Opnieuw wordt eerst de parameter I op de stapel gezet.
[3]Vervolgens wordt I met 1 verhoogd.
[4]De waarde van I wordt op het scherm getoond.
[5]Dan volgt een vergelijking waarin gekeken wordt of I kleiner is dan 10. Als dit zo is, wordt het terugkeeradres op de stapel gezet en wordt gesprongen naar de procedure Opnieuw. Daar wordt de parameter I weer op de stapel gezet en met 1 verhoogd. In de vergelijking wordt weer gekeken of I kleiner dan 10 is. Zolang I kleiner is dan 10, wordt de functie Opnieuw aangeroepen. Als I de waarde 10 bereikt heeft, wordt niet meer naar de procedure Opnieuw gesprongen en wordt deze beëindigd door nogmaals de waarde van I op het scherm te zetten. De procedure is nu afgewerkt, en er kan teruggekeerd worden naar het terugkeeradres. Omdat de procedure Opnieuw door zichzelf werd aangeroepen, is het terugkeeradres de opdracht die volgt op de sprongopdracht.

Op het moment dat die sprongopdracht gegeven wordt, heeft I de waarde 9. Nadat deze waarde naar het scherm geschreven is, wordt weer teruggekeerd naar het terugkeeradres, waar I de waarde 8 heeft. Dit gaat zo door tot we naar het terugkeeradres in het hoofdprogramma springen. Op deze manier ontstaat er een rij getallen, eerst van 1 tot en met 10 en vervolgens van 10 tot en met 1.

Recursie is wellicht een verwarrend onderwerp. Het is echter nodig dat je het principe begrijpt, omdat het nogal eens gebruikt wordt bij sorteermethoden en bij het groeperen van gegevens. In het hoofdstuk over sorteren zal recursie dus weer aan de orde komen.

6.9 Een echt programma

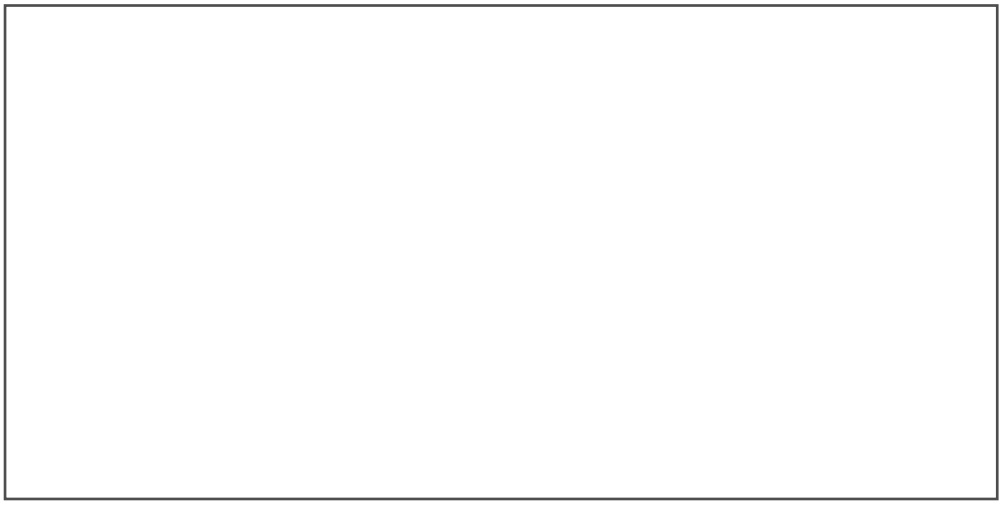
Tot nu toe hebben we steeds kleine programma's gemaakt. We zijn er nu wel aan toe om het geleerde toe te passen in een groot programma.

We gaan een programma maken waarbij de gebruiker kan opgeven welke tafel van vermenigvuldiging hij of zij wil zien. Na de opgave moet de tafel op het beeldscherm getoond worden en moet het programma vragen of de gebruiker nog meer tafels wil zien. Als deze vraag met een druk op de letter "J" of "j" beantwoord wordt, dan moet een nieuwe tafel opgegeven kunnen worden. Als de letter "N" of "n" wordt

ingedrukt, dan bedankt het programma de gebruiker en wordt het programma afgebroken:

```
PROGRAM TAFELS;
USES CRT;

   PROCEDURE ZetAchtergrond;
   BEGIN
      TextBackground(7);
      ClrScr;
      Window(5,2,75,2);
      TextBackground(0);
      TextColor(15);
      ClrScr;
      GotoXY(10,1);
      Write('T A F E L S   V A N   ');
      Write('V E R M E N I G V U L D I G I N G');
      Window(8,7,35,19);
      TextBackground(1);
      ClrScr
   END;

   FUNCTION GekozenTafel:Word;
   VAR
      TAFEL: Word;

   BEGIN
      Window(40,5,77,8);
      TextBackground(1);
      TextColor(15);
      ClrScr;
      GotoXY(2,2);
      Write('Geef een tafel op');
      GotoXY(2,3);
      Write('en druk op de Enter-toets... ');
      Readln(TAFEL);
      TextBackground(7);
      ClrScr;
      GekozenTafel := TAFEL
   END;

   FUNCTION Vermenigvuldiging(A,B:Longint):Longint;
   BEGIN
      Vermenigvuldiging := A * B
   END;

   PROCEDURE DrukTafelAf;
   VAR
      I    : Byte;
      TAFEL: Word;
   BEGIN
      TAFEL := GekozenTafel;
      Window(8,7,35,19);
      TextBackground(1);
```

```
      TextColor(15);
      ClrScr;
      FOR I := 1 TO 10 DO
      BEGIN
         GotoXY(5,I+1);
         Write(I:2,'   x  ',TAFEL:5,' = ',
               Vermenigvuldiging(I,TAFEL):6)
      END
   END;

   FUNCTION NogMeer:Char;
   VAR
      LETTER: Char;
   BEGIN
      Window(40,18,78,20);
      TextBackground(1);
      TextColor(15);
      ClrScr;
      REPEAT
         GotoXY(2,2);
         Write('Wilt U nog meer tafels doen? J/N');
         LETTER := Readkey;
      UNTIL LETTER IN ['J','j','N','n'];
      TextBackground(7);
      ClrScr;
      NogMeer := LETTER
   END;

   PROCEDURE Bedank;
   BEGIN
      Window(1,1,80,25);
      TextBackground(0);
      TextColor(15);
      ClrScr;
      GotoXY(15,10);
      Write('Leuk dat u TAFELS hebt gebruikt. ');
      Write('Bedankt en tot ziens!');
      Readln
   END;

BEGIN
   ZetAchtergrond;
   REPEAT
      DrukTafelAf;
   UNTIL NogMeer IN ['N','n'];
   Bedank
END.
```



Afbeelding 7

Regels:Toelichting:

[1]3-17Procedure ZetAchtergrond.
5-6Zet de achtergrondkleur op lichtgrijs en veeg het scherm schoon.
7-13Maak een zwart venster bovenin het scherm. Zet daarin in witte letters dat het om de tafels van vermenigvuldiging gaat.
14-16Maak een blauw venster aan de linkerkant van het scherm voor het afdrukken van de tafels.
[3]18-34Function GekozenTafel.
19-20 Declareer een lokale variabele TAFEL.
22-29Maak een blauw venster rechts bovenin het scherm, waarin gevraagd wordt een tafel op te geven.
30 Lees met het toetsenbord een waarde in de lokale variabele TAFEL.
31-32Maak het venster onzichtbaar.
33Geef de functie GekozenTafel de waarde die in de lokale variabele TAFEL staat.
[4] 35-38Function Vermenigvuldiging.
37Vermenigvuldig de parameters A en B met elkaar en geef de functie Vermenigvuldiging de waarde van deze uitkomst.
[2]39-55Procedure DrukTafelAf.
40-42Declareer de lokale variabelen I van het type Byte en TAFEL van het type Word.
[3]44Zet de waarde van de functie GekozenTafel in de lokale variabele TAFEL.
45-48Zet een blauw venster op de plaats die in ZetAchtergrond al gebruikt is en veeg dit venster schoon.
[4]49-54Ga een FOR-lus in die tien keer doorlopen wordt en plaats na elke keer een regel van de gevraagde tafel op het scherm.
[5]56-72 Function NogMeer.
57-58Declareer een lokale variabele LETTER van het type Char.
60-68Maak een blauw venster rechts onderin het scherm en vraag of er nog meer tafels getoond moeten worden. Lees het letterteken dat door de gebruiker ingetoetst is in de lokale variabele LETTER.
69-70Maak het venster onzichtbaar.
71Geef de functie NogMeer de waarde die in LETTER staat.
[6] 73-83Procedure Bedank.
75-82Maak een venster ter grootte van het hele scherm, zet de achtergrond op zwart en de tekst op wit. Veeg het scherm schoon en zet midden op het scherm een bedankje.

84-90Hoofdprogramma.

[1]85 Roep de procedure ZetAchtergrond aan.

[2]86-88Ga een REPEAT-lus in die duurt tot de functie NogMeer een "N" of een "n" retourneert. Roep in de lus telkens de procedure DrukTafelAf aan.

[6]89Roep als uit de REPEAT-lus gesprongen is de procedure Bedank aan en beëindig het programma.

Toelichting:

[1]Het programma begint met een aanroep van de procedure ZetAchtergrond. TextBackground wordt hier op 7 gezet (lichtgrijs) en het scherm wordt schoongeveegd.
Vervolgens wordt in de procedure Window bovenin het scherm een smalle balk gemaakt. De achtergrond van deze balk is zwart. Met de procedure Write wordt vervolgens de titel van het programma met witte letters in de balk weergegeven. Omdat we vast willen aangeven waar straks de tafels worden getoond, wordt er links nu al een blauw, leeg venster gemaakt.

[2]De REPEAT UNTIL-lus die hier wordt ingegaan, duurt tot er vanaf het toetsenbord een "N" of een "n" ingetoetst wordt, ten teken dat de gebruiker het programma wil beëindigen. Na iedere keer dat de lus is doorlopen, wordt de procedure DrukTafelAf aangeroepen.

[3]Na het betreden van de procedure DrukTafelAf, wordt allereerst de functie GekozenTafel aangeroepen. In deze functie wordt rechts bovenin het scherm een blauw venster gemaakt waarin de gebruiker uitgenodigd wordt om op te geven welke tafel hij of zij wil zien. Met Readln wordt een waarde vanaf het toetsenbord ingelezen in de lokale variabele TAFEL. Daarna wordt het venster onzichtbaar gemaakt door het lichtgrijs te kleuren (de kleur van de achtergrond).

[4] Nu wordt er een FOR-lus ingegaan die tien keer doorlopen wordt. Na iedere keer dat de lus doorlopen is, wordt I met 1 verhoogd. In de GotoXY-opdracht wordt "I+1" gebruikt om aan te geven op welk regelnummer een regel van de tafel moet verschijnen. Aan de procedure Write wordt eerst I doorgegeven en daarna de string "x". Vervolgens wordt de lokale variabele TAFEL doorgegeven, opnieuw gevolgd door een string: "=".
Als laatste worden de waarden van de functie Vermenigvuldiging opgenomen in de gegevens die doorgegeven worden aan de procedure Write. De functie Vermenigvuldiging heeft twee parameters van het type Longint. De variabelen I en TAFEL, waarvan de waarden in deze parameters worden doorgegeven, zijn respectievelijk van het type Byte en Word. De maximale waarde die in een type Word past is 65535. Stel dat je opgeeft dat je de tafel van 65535 wilt zien. Dan past 2×65535 al niet meer in een variabele van het type Word. Vandaar dat de parameters van de functie Vermenigvuldiging van het type Longint zijn. De functie Vermenigvuldiging retourneert ook een variabele van het type Longint.

[5]Nadat de tafel is afgedrukt, wordt op regel 88 de REPEAT-lus afgesloten met een UNTIL. Doordat er staat: "UNTIL Nogmeer IN ..." wordt telkens de functie NogMeer aangeroepen. De functie NogMeer maakt eerst met behulp van de procedure Window rechts onderin het scherm een blauw venstertje. In dit venster wordt gevraagd of er nog meer tafels getoond moeten worden. De gebruiker wordt uitgenodigd om een "J" of een "N" in te tikken.
Omdat ook hier gebruik is gemaakt van een REPEAT UNTIL-lus, wordt een ander letterteken dan een "J" of "j" en een "N" of "n" niet geaccepteerd. Met behulp van de functie Readkey wordt een letterteken vanaf het toetsenbord ingelezen. De gebruiker hoeft dit letterteken niet te bevestigen met een druk op de Enter-toets, omdat de functie Readkey bestemd is om één letter van het toetsenbord te lezen. Als deze letter gelezen is, wordt er gekeken of het een "J" of "j" of een "N" of "n" betreft. Als dit niet het geval is, blijven we in de lus en kan er opnieuw een letter worden ingelezen. Dit gaat net zolang door tot een van de juiste letters wordt ingetikt.

Als in de REPEAT-lus in het hoofdprogramma de functie NogMeer een "J" of een "j" retourneert, wordt de lus opnieuw doorlopen en wordt de functie DrukTafelAf aangeroepen. Als de functie NogMeer een "N" of een "n" teruggeeft, moet het programma beëindigd worden en springen we uit de lus.

[6]Tenslotte roepen we de procedure Bedank aan. De procedure Bedank maakt het hele scherm zwart en

plaatst een bedankje in het midden.

6.10 Opgaven

1. **Maak een programma dat een stukje tekst in een string leest en dit naar een functie stuurt die het in omgekeerde volgorde retourneert. Zet beide teksten onder elkaar op het scherm.**

2. Maak een programma dat een datum naar een functie stuurt. De functie moet de dag uit de datum isoleren en deze in numerieke vorm retourneren. De datum heeft het formaat "DD-MM-JJJJ". De functie moet met meerdere scheidingstekens kunnen werken. Als op de eerste positie van de datum geen cijfer staat, moet de functie een 0 retourneren.