16. Objecten

16.1 Inleiding

De wijze van programmeren die we tot nu toe gevolgd hebben, heet gestructureerd programmeren: je verdeelt een gesteld probleem in stukjes, zodat je een onderverdeling krijgt in procedures en functies. Het deelprobleem verdeel je weer in stukjes, net zolang tot je het probleem stapsgewijs op kunt lossen. Pascal is zelfs oorspronkelijk ontwikkeld om studenten te dwingen tot gestructureerd programmeren. Voor de gegevens waar het programma mee werkt, wordt een gegevensstructuur ontwikkeld die los staat van de procedures en functies.

Vanaf versie 5.5 heeft Borland de faciliteiten voor objectgericht programmeren in de taal ingebouwd. Bij objectgericht programmeren gaat men ervan uit dat gegevens en de bewerking van die gegevens bij elkaar horen. Uitgaande van een object worden de gedefinieerde functies en procedures de methoden van het object genoemd. De methoden van het object voeren bewerkingen op de gegevens van het object uit. De gegevens van het object worden alleen benaderd via de methoden van het object.

Bij gestructureerd programmeren komt het nog wel eens voor dat een stuk programma heel erg lijkt op een ander stuk programma. In zo'n geval moeten, ondanks dat er slechts kleine afwijkingen zijn, hele stukken programmacode worden herschreven. Een groot voordeel van objectgericht programmeren is dat alleen dat wat veranderd is, herschreven moet worden. De rest van de programmacode kun je hergebruiken. Het mechanisme dat hiervoor gebruikt wordt, heet "overerving". Objecten kunnen namelijk "kinderen" krijgen die alle eigenschappen van deze "ouder" erven. Hiervoor wordt meestal de Engelse term "inheritance" gebruikt. In de op die manier ontstane nieuwe objecten kun je de geërfde eigenschappen veranderen, of er nieuwe eigenschappen aan toevoegen. Ook de gegevensstructuur wordt van de ouder geërfd. In het geërfde object kun je werken met de gegevensstructuur van de ouder en je kunt deze gegevensstructuur ook uitbreiden.

Om objectgericht te programmeren moet je een andere denktrant ontwikkelen dan je gewend was bij het gestructureerd programmeren. Je moet de gegevensstructuur van een programma en de bewerking die deze gegevens moeten ondergaan, in overzichtelijke onderdelen verdelen. Deze onderdelen zijn de objecten van het programma. Als je eenmaal gewend bent aan de denkwijze die vereist is voor het werken met objecten, dan is het objectgericht programmeren een natuurlijkere werkwijze dan het gestructureerd programmeren.

16.2 Werken met objecten

De declaratie van een object lijkt erg veel op de declaratie van een record. In de nu volgende demonstratie van het gebruik van objecten, wordt eerst een object gedeclareerd dat een

demonstratie geeft van de in Turbo Pascal beschikbare wiskundige functies. Het wiskunde-object moet vragen om de invoer van een getal en moet vervolgens de verschillende bewerkingen van dat getal op het scherm laten zien.

Als we nu naast de mogelijkheden die het wiskunde-object biedt, ook nog de mogelijkheid willen hebben om het ingevoerde getal te gebruiken voor machtsverheffen, dan heeft het object dus behoefte aan invoer van een tweede getal. Dit nieuwe getal moet gebruikt kunnen worden als exponent voor het machtsverheffen.

Een programma dat volgens de gestructureerde programmeertechniek ontwikkeld is, zal grotendeels herschreven moeten worden als we het met een nieuwe taak willen uitbreiden. Er komt dan een uitbreiding op de gegevensstructuur, er moet een nieuwe functie Machtsverheffen gemaakt worden en de in- en uitvoerprocedures moeten herschreven worden.

Bij een objectgerichte aanpak is dit niet nodig. Er kan bij deze methode eenvoudigweg een object gedefinieerd worden dat erfgenaam is van het wiskunde-object. Dit nieuwe object erft alle eigenschappen van de ouder. Er wordt een veld Exponent toegevoegd en een functie Machtsverheffen. Daarnaast wordt de procedure voor het invoeren van een Exponent toegevoegd en wordt de uitvoer bijgewerkt. Dit voorgaande betekent dat we slechts enkele regels code hoeven toe te voegen. Voor het overige kunnen we gewoon de code van het wiskunde-object gebruiken.

Het wiskunde-object wordt ondergebracht in een unit WISKUNDE:

```
{$X+}
UNIT Wiskunde;
Interface
USES CRT, DIVERSEN;

TYPE
TWiskunde = Object
   GETAL: Real;
   PROCEDURE Init;
   FUNCTION Kwadraat:Real;
   FUNCTION Vierkantswortel:Real;
   FUNCTION Sinus:Real;
   FUNCTION BoogTangens:Real;
   FUNCTION Cosinus:Real;
   FUNCTION HeelGetal:Integer;
   FUNCTION Fractie:Real;
   FUNCTION NLogaritme:Real;
   PROCEDURE Titel;
   PROCEDURE Schrijf;
END;

CONST
   WACHT: Boolean = True;
Implementation

PROCEDURE TWiskunde.Init;
VAR
   SGETAL: String;
   CODE  : Integer;

BEGIN
   Venster(VX1,VY1,VX2,VY2,4,14);
   Titel;
   REPEAT
      GotoXY(5,5);
      ClrEol;
      Write('Voer getal in s.v.p ');
      Readln(SGETAL);
      Val(SGETAL,GETAL,CODE);
      IF CODE <> 0 THEN
      BEGIN
         GotoXY(10,8);
         Write(#7'Een getal svp!!!');
         ReadKey;
         GotoXY(10,8);
         ClrEol
```

```
      END
   UNTIL CODE = 0
END;

FUNCTION TWiskunde.Kwadraat:Real;
BEGIN
   Kwadraat := SQR(GETAL)
END;

FUNCTION TWiskunde.Vierkantswortel:Real;
BEGIN
   Vierkantswortel := SQRT(GETAL)
END;

FUNCTION TWiskunde.Sinus:Real;
BEGIN
   Sinus := Sin(GETAL)
END;

FUNCTION TWiskunde.BoogTangens:Real;
BEGIN
   BoogTangens := ArcTan(GETAL)
END;

FUNCTION TWiskunde.Cosinus:Real;
BEGIN
   Cosinus := Cos(GETAL)
END;

FUNCTION TWiskunde.NLogaritme:Real;
BEGIN
   NLogaritme := Ln(GETAL)
END;

FUNCTION TWiskunde.HeelGetal:Integer;
BEGIN
   HeelGetal := Trunc(GETAL)
END;

FUNCTION TWiskunde.Fractie:Real;
BEGIN
   Fractie := Frac(GETAL)
END;

PROCEDURE TWiskunde.Titel;
VAR
```

```
  ZIN: String;

BEGIN
  ZIN := ' WISKUNDE TURBO PASCAL ';
  Window(VX1,VY1,VX2,VY2);
  GotoXY(((VX2-VX1) DIV 2) - (Length(ZIN) DIV 2),1);
  Write(ZIN);
  Window(VX1+2,VY1+1,VX2-2,VY2-1)
END;

PROCEDURE TWiskunde.Schrijf;
BEGIN
   Venster(VX1,VY1,VX2,VY2,3,0);
   Titel;
   GotoXY(4,1);
   Write('Ingevoerd getal          : ',GETAL:6:5);
   GotoXY(4,2);
   Write('Wetenschappelijke notatie: ',GETAL);
   GotoXY(4,3);
   Write('Het integere deel is     : ',HeelGetal:6);
   GotoXY(4,4);
   Write('Fractie                  : ',Fractie:6:5);
   GotoXY(4,5);
   Write('Sinus                    : ',Sinus:6:5);
   GotoXY(4,6);
   Write('Cosinus                  : ',Cosinus:6:5);
   GotoXY(4,7);
   Write('BoogTangens              : ');
   Write(BoogTangens:6:5);
   GotoXY(4,8);
   Write('Natuurlijk logaritme     : ');
   Write(NLogaritme:6:5);
   IF WACHT THEN ReadKey
END;

END.
```

Regels:Toelichting:

[1]1 Zet de optie {$X+} aan.
4 Gebruik de units CRT en DIVERSEN.
[2]5-19Declareer een object TWiskunde.
[3]20-21Declareer een getypeerde constante van het type Boolean.
23-45Procedure TWiskunde.Init.
24-26Declareer de benodigde lokale variabelen.
[4]28 Roep de procedure Venster aan.
[5]29 Roep de procedure TWiskunde.Titel aan.
[7]30-44Voer een getal in en zet dat in TWiskunde.Getal.
[8]46-77 Diverse wiskundige functies.
[5]78-87Procedure TWiskunde.Titel.
79-80Declareer een lokale variabele ZIN.
[6]82 Zet de titel van het scherm in ZIN.
83 Vergroot het geldige venster.
84-85 Zet de titel gecentreerd in het frame van het venster.
86 Geef het venster weer z'n oorspronkelijke afmetingen.
88-111 Procedure TWiskunde.Schrijf.
[9]90 Roep de procedure Venster aan.
[10]91 Roep de procedure TWiskunde.Titel aan.
[11]92-109 Schrijf de uitkomsten van de verschillende functies naar het scherm.
[12]110 Als WACHT True is, wacht dan op het indrukken van een toets.

Toelichting:

[1]Om functies aan te kunnen roepen alsof het procedures zijn moet de optie {$X+} geplaatst worden.

[2]De declaratie van een object lijkt erg veel op de declaratie van een record. In plaats van het beschermde woord "Record" wordt nu bij de type-definitie het beschermde woord "Object" gebruikt. Een record bevat alleen velden van een bepaald type, terwijl een object naast dat soort velden, ook procedures en functies kan bevatten. De procedures en functies worden samen de methoden van het object genoemd. Binnen een object worden eerst de verschillende velden gedeclareerd, gevolgd door de methoden.

Als achter de methoden het beschermde woord "Private" wordt gezet, dan kan er weer een blok van gegevens en methoden gedeclareerd worden. Dit blok is dan alleen toegankelijk vanuit het object en vanuit andere objecten die in dezelfde unit zijn gedeclareerd. De typering van objecten

gebeurt in een programma of in een unit. Objecten kunnen niet in een procedure of functie gedeclareerd worden.

Een object wordt in het interface-gedeelte van de unit gedeclareerd. De uitwerking van de methoden gebeurt in het implementatie-deel.

[3]De variabele WACHT hebben we nodig om te kunnen bepalen wanneer er gewacht moet worden op het indrukken van een toets.

[4]De procedure Init moet het getal lezen dat gebruikt wordt om aan de beschikbare wiskundige functies door te geven. We hebben dus een venstertje nodig en roepen de procedure Venster uit de unit DIVERSEN aan.

[5]Vervolgens roepen we de procedure TWiskunde.Titel aan. Binnen het object kunnen we volstaan met het aanroepen van Titel. Buiten het object is dat anders. Als we een variabele Wiskunde van het type TWiskunde hebben, dan wordt een methode aangeroepen door achter de naam van de variabele een punt te zetten, gevolgd door de naam van de methode. In dit geval zou dat Wiskunde.Init kunnen zijn. Dit werkt dus op precies dezelfde manier als het bereiken van velden in een record. Ook het WITH-statement wordt met objecten op dezelfde manier gebruikt als met records.

[6]De procedure Titel schrijft het bij het object behorende opschrift gecentreerd in het frame van het venster. Dit is weer zo'n klein trucje. Door van X2 de waarde van X1 af te trekken en het resultaat door twee te delen, wordt het midden van het venster bepaald. Vervolgens wordt de helft van de lengte van de zin van het gevonden midden afgetrokken. Het resultaat is de X-coördinaat voor de GotoXY-opdracht. De titel verschijnt nu in het midden van het frame.

[7]Met een string wordt een getal ingelezen. De string wordt door VAL geconverteerd naar het veld TWiskunde.Getal. De foutafhandeling is reeds genoegzaam behandeld.

[8]Turbo Pascal beschikt over een aantal wiskundige functies die door de methoden van TWiskunde aangeroepen worden:

Functie:	Bewerking:	Resultaat:
ArcTan(X:Real)	Berekent de boogtangens van X in radialen.	Real
Cos(X:Real)	Berekent de cosinus van X.	Real
Exp(X:Real)	Berekent de exponent van X.	Real
SQR(X:Real)	Berekent het kwadraat van X.	Type van X
SQRT(X:Real)	Berekent de vierkantswortel van X.	Real
Ln(X:Real)	Berekent de natuurlijke logaritme van X.	Real

[9]De procedure Schrijf roept weer eerst de procedure Venster aan om een venster te maken waar de resultaten in geschreven moeten worden.

[10]Titel wordt aangeroepen om de titel in het frame te plaatsen.

[11]Dan worden alle functies van het object aangeroepen en worden de uitkomsten in het venster geschreven.

[12]Als de getypeerde constante WACHT True is, stopt de uitvoering van het programma tot er een toets wordt ingedrukt.

Nu we de unit met het wiskunde-object gemaakt hebben, moeten we een programma maken dat met dit object gaat werken. Dit programma is vrij kort:

```
PROGRAM OBJECT_1;
USES CRT, WISKUNDE;

VAR
   WISKUNDE_OBJECT: TWiskunde;
BEGIN
   TextBackground(0);
   ClrScr;
   WISKUNDE_OBJECT.Init;
   WISKUNDE_OBJECT.Schrijf
END.
```

Regels:Toelichting:

[1]2 Gebruik de units CRT en WISKUNDE.
[2]3-4Declareer een variabele van het type TWiskunde.
6-7 Maak het scherm zwart en veeg het schoon.
[3]8 Roep de methode WISKUNDE_OBJECT.Init aan.
[4]9 Roep de methode WISKUNDE_OBJECT.Schrijf aan.

Toelichting:

[1]Door de unit WISKUNDE in de USES-regel op te nemen, kunnen we het object TWiskunde gebruiken.

[2]De variabele WISKUNDE_OBJECT krijgt als typering TWiskunde.

[3]De methode Init van het object wordt bereikt met WISKUNDE_OBJECT.Init. Dus eerst de naam van de variabele, gevolgd door een punt, en dan de naam van de methode.

[4]De methode Init heeft een getal ingelezen. We roepen de methode WISKUNDE_OBJECT.Schrijf aan om een rapportage te verzorgen.

16.3 Overerving

In het nu volgende programma zullen de voordelen van het objectgericht programmeren heel duidelijk worden. In het object TWiskunde riepen de methoden de verschillende wiskundige Turbo Pascal-functies aan. Turbo Pascal kent geen ingebouwde functie voor machtsverheffen. Die functie zou je zelf moeten maken. De functie Machtsverheffen gebruikt dan weer de beschikbare ingebouwde functies.

Voor machtsverheffen heb je behalve een al dan niet gebroken getal, een exponent nodig. De exponent geeft aan tot welke macht het getal verheven moet worden. Het object moet nu dus niet alleen vragen om de invoer van een getal, maar ook om de invoer van een exponent. In de rapportage moet niet alleen de uitkomst van het machtsverheffen staan, maar ook de overige wiskundige bewerkingen.

In de unit MACHT wordt het fenomeen overerving gedemonstreerd. Ook wordt getoond hoe objecten dynamisch gedeclareerd kunnen worden. De titel die in het frame geplaatst wordt, wordt hier aangepast:

UNIT MACHT;

```
Interface
USES CRT, DIVERSEN, WISKUNDE;

TYPE
PMacht = ^TMacht;
TMacht =  Object(TWISKUNDE)
   Exponent: Integer;
   CONSTRUCTOR Init;
   FUNCTION Machtsverheffen: Real;
   PROCEDURE Titel;
   PROCEDURE Schrijf;
   DESTRUCTOR Done;
END;

Implementation
CONSTRUCTOR TMacht.Init;
VAR
   SGETAL: String;
   CODE  : Integer;

BEGIN
   TWiskunde.Init;
   GotoXY(5,7);
   Write('Geef een exponent voor machtsverheffen: ');
   REPEAT
      GotoXY(35,7);
      ClrEol;
      Readln(SGETAL);
      Val(SGETAL,Exponent,CODE);
      IF CODE <> 0 THEN
      BEGIN
         GotoXY(10,9);
         Write(#7,'Een geheel getal svp!');
         ReadKey
      END
   UNTIL CODE = 0
END;

FUNCTION TMacht.Machtsverheffen;
BEGIN
   Machtsverheffen := Exp(NLogaritme * Exponent)
END;

PROCEDURE TMacht.Titel;
VAR
```

```
   ZIN: String;

BEGIN
   ZIN := ' WISKUNDE TURBO PASCAL + MACHTSVERHEFFEN ';
   Window(VX1,VY1,VX2,VY2);
   GotoXY(((VX2-VX1) DIV 2) - (Length(ZIN) DIV 2),1);
   Write(ZIN);
   Window(VX1+2,VY1+1,VX2-2,VY2-1)
END;

PROCEDURE TMacht.Schrijf;
BEGIN
   WACHT := False;
   Inherited Schrijf;
   GotoXY(5,9);
   Write(GETAL:6:5,' tot de macht ',Exponent,' = ');
   Write(Machtsverheffen:6:5);
   ReadKey;
END;

DESTRUCTOR TMacht.Done;
BEGIN
END;

END.
```

Regels:Toelichting:

```
3           Gebruik de units CRT, DIVERSEN en WISKUNDE.
  [1]4-13     Declareer een object als erfgenaam van TWiskunde.
[2]15-35      Constructor TMacht.Init.
[3]20         Roep TWiskunde.Init aan.
  [4]21-34    Lees een geheel getal in.
[5]36-39      Function TMacht.Machtsverheffen.
  [6]40-49    Function TMacht.Titel.
 50-58        Procedure TMacht.Schrijf.
[7]52         Zet de variabele WACHT op False.
  [8]53       Roep TWiskunde.Schrijf aan.
  [9]54-56Zet de uitkomst van de functie machtsverheffen op de
          onderste regel van het venster.
59-61         Destructor TMacht.Done.
```

Toelichting:

[1]De unit van het nieuwe object TMacht is aanmerkelijk kleiner dan de unit van TWiskunde. Toch kan TMacht meer dan TWiskunde. Met de volgende declaratie:

TMacht = Object(TWiskunde)

heeft het object TMacht alle eigenschappen van TWiskunde geërfd. TMacht beschikt zowel over het veld GETAL als over de in TWiskunde gedeclareerde methoden.

In de declaratie van TMacht komen drie namen voor die ook in TWiskunde voorkwamen:

Init
Titel
Schrijf

Bovendien heeft TMacht een nieuwe functie Machtsverheffen. De methoden die al voorkwamen, worden in TMacht overschreven. Het overschrijven van methoden is heel belangrijk voor het aanbrengen van veranderingen in de geërfde eigenschappen. Met andere woorden: voor overschreven methoden worden nieuwe methoden gedeclareerd. Deze nieuwe methoden kunnen al dan niet gecombineerd worden met de oorspronkelijke methoden.

In de declaratie wordt PMacht gedeclareerd als pointer naar TMacht. Een variabele van het type PMacht zal dus een pointer zijn naar een object van het type TMacht, dat zich op de heap bevindt.

[2]Bij de Init-methode van het object is weer iets nieuws te zien. Hier is de aanduiding "Procedure" vervangen door de aanduiding "Constructor". Constructor en destructor zijn begrippen die samenhangen met objectgericht programmeren. Deze begrippen komen in de volgende paragraaf aan de orde. Ga er voorlopig maar van uit dat constructors en destructors net als gewone procedures werken.

[3]Als eerste wordt TWiskunde.Init aangeroepen. Omdat TMacht een erfgenaam van TWiskunde is, kun je vanuit de methoden van het object de methoden van de ouder aanroepen. Door deze aanroep maakt de ouder een venster aan en vraagt om invoer van een getal.

[4]Bij terugkomst wordt er een waarde in het veld TMacht.Exponent gelezen. Init was een overschreven methode. Door de Init-methode van de ouder aan te roepen, worden beide methoden verenigd tot één uitgebreide methode.

[5]In TMacht.Machtsverheffen wordt de Turbo Pascal-functie Exp aangeroepen. Hier wordt als parameter de natuurlijke logaritme van TMacht.Getal vermenigvuldigd met de waarde die in het veld TMacht.Exponent werd meegegeven.

[6]De procedure TMacht.Titel roept niet de methode van de ouder aan. Het is dus een eigen methode van TMacht. Evenals de Titel-methode van de ouder zet deze procedure een opschrift in het frame.

[7]In TMacht.Schrijf wordt eerst de getypeerde constante WACHT op False gezet.

[8]Nu wordt de procedure Schrijf van de ouder aangeroepen. In dit geval gebruiken we niet TWiskunde.Schrijf, maar Inherited.Schrijf. Het beschermde woord "Inherited", gevolgd door de naam van de methode, leidt tot hetzelfde resultaat als het aanroepen van de methode via de typering. Deze manier van aanroepen is beschikbaar vanaf versie 7.0.

[9]Omdat WACHT op False gezet is, wacht de methode TWiskunde.Schrijf niet op het indrukken van een toets, maar keert de methode na het tonen van de resultaten terug. Nu wordt de uitkomst van de functie Machtsverheffen onderaan in het venster geschreven.

Ook voor het werken met de unit MACHT is maar een klein programma nodig. In de volgende paragraaf "Virtuele methoden" wordt het volgende programma geoptimaliseerd:

```
PROGRAM OBJECT_2;
USES CRT, MACHT;

VAR
   MachtObject: PMacht;
BEGIN
   TextBackground(0);
   ClrScr;
   MachtObject := New(PMacht,Init);
   WITH MachtObject^ DO Schrijf;
   Dispose(MachtObject,Done)
END.
```

Regels:Toelichting:

```
[1]4   Declareer een dynamische variabele MachtObject.
[2]8   Wijs ruimte op de heap toe aan MachtObject en  roep  de
       methode MachtObject.Init aan.
[3]9        Roep de methode MachtObject.Schrijf aan.
[4]10       Geef de ruimte op de heap weer vrij.
```

Toelichting:

[1]Door als typering PMacht te gebruiken, wordt de variabele MachtObject een dynamische variabele.

[2]Net als voor iedere andere dynamische variabele, moet er ruimte op de heap toegewezen worden. Voor objecten is de Turbo Pascal-procedure New uitgebreid. New krijgt hier als parameter het type van het object en de naam van de constructor die aangeroepen moet worden mee. In dit geval is dat de constructor Init. New wijst nu het geheugen toe en roept daarna de doorgekregen constructor aan.

[3]Teruggekomen uit de constructor Init, kunnen we de methode MachtObject.Schrijf aanroepen. Hier kun je zien dat het WITH-statement op dezelfde manier gebruikt kan worden als bij records.

[4]Als het object niet meer nodig is, kan de ruimte op de heap weer vrijgegeven worden. Hiervoor is de Turbo Pascal-procedure Dispose uitgebreid. Aan deze procedure wordt de naam van de variabele doorgegeven, samen met de naam van de destructor. In dit geval is dat de destructor Done.

16.4 Virtuele methoden

Als het programma OBJECT_2 gedraaid wordt, dan werkt het programma niet helemaal goed. De procedure Titel is in TMacht overschreven. Het nieuwe opschrift moet ons laten weten dat dit object ook kan machtsverheffen. OBJECT_2 zweert echter bij de oude titel en verandert het opschrift niet.

Dit fenomeen wordt veroorzaakt doordat de procedure Titel aangeroepen wordt vanuit methoden die behoren bij het object TWiskunde. TWiskunde kent helemaal geen procedure TMacht.Titel. Op deze manier heb je natuurlijk niet veel aan het erven van methoden. Als vanuit een geërfde methode een overschreven methode wordt aangeroepen, krijg je de verkeerde. Dit probleem wordt veroorzaakt door de compiler. Tijdens het compileren vervangt

deze de benaming van methoden door adressen. Het aanroepen van de procedure Titel is dus eigenlijk een sprongopdracht naar een adres. In dit geval is dat het verkeerde adres. Het toewijzen van een adres tijdens het compileerproces heet "vroege binding".

Om dit soort situaties te voorkomen, kunnen we bij objectgericht programmeren gebruik maken van "late binding". Dit wil zeggen dat het adres voor de sprongopdracht niet wordt toegewezen tijdens het compileren, maar tijdens de uitvoering van het programma. Als je late binding wilt gebruiken, moet de eerste aanroep van een object altijd een constructor zijn. Een object kan meerdere constructors hebben. Het is een goede gewoonte om voor constructors en destructors steeds dezelfde benamingen te hanteren. Voor een constructor wordt meestal het woord "Init" gebruikt, en voor een destructor "Done". Het hanteren van gelijke benamingen maakt het makkelijker om objecten te initiëren en weer af te breken.

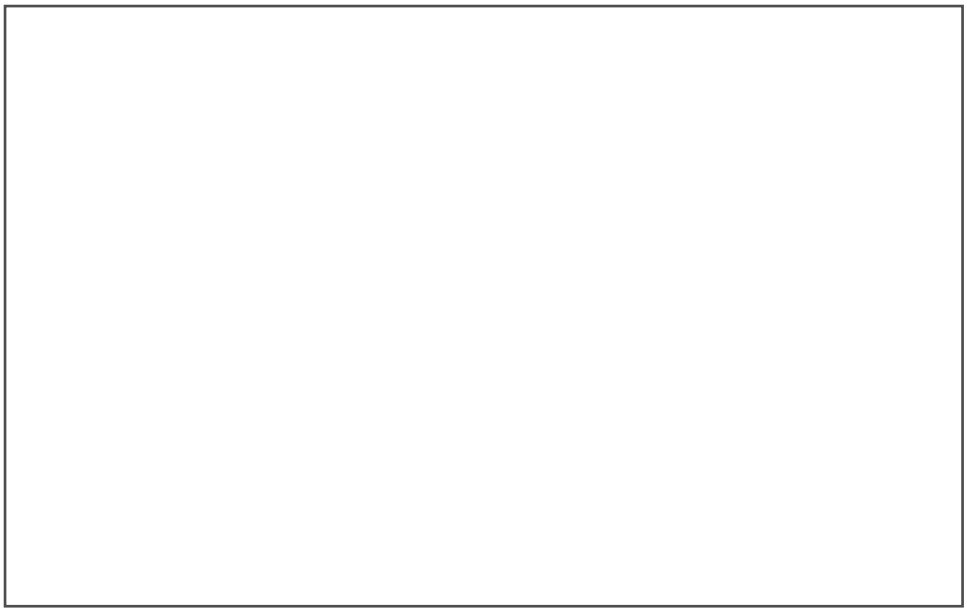
Een constructor bouwt een tabel met adressen van virtuele methoden op. Een virtuele methode is een procedure of functie die bij de declaratie het beschermde woord "Virtual" toegevoegd krijgt. De tabel met virtuele methoden wordt ook wel VMT (Virtual Method Table) genoemd. Het aanroepen van een virtuele methode gebeurt nu niet via een door de compiler verstrekt adres, maar door het zoeken van het juiste adres in de VMT.

Zoals een aanroep van de constructor een VMT opbouwt, zo breekt het aanroepen van een destructor de VMT weer af. Hiermee zijn dan tegelijk de nieuwe begrippen constructor en destructor toegelicht.

Tot slot moeten we het programma OBJECT_2 nog correct laten draaien. Hiervoor moet in de declaratie van TWiskunde achter de declaratie van de procedure Titel het statement "Virtual;" worden toegevoegd. Dus:

Procedure Titel;Virtual;

Bij de declaratie van TMacht moet hetzelfde gedaan worden. Als je het programma nu draait, wordt de juiste titel getoond:



Afbeelding 15

16.5 Turbo Vision

Vanaf Turbo Pascal 6.0 levert Borland Turbo Vision mee. Dit is een professionele gereedschapskist en een samenhangend systeem voor objectgericht programmeren.

Turbo Vision geeft je de mogelijkheid om programma's te bouwen die zich op dezelfde manier manifesteren als de ontwikkelomgeving van Turbo Pascal. Deze ontwikkelomgeving is zelfs met Turbo Vision gebouwd.

Als je zelf een programma maakt, gaat er veel tijd zitten in het ontwikkelen van menu's, invoerschermen en wat al niet meer. Dit is kostbare tijd die dan niet besteed wordt aan het probleem waarvoor het programma gemaakt wordt. Borland vergelijkt in de handleiding de schermen en opslagstructuren met het skelet van het programma, en het inhoudelijke deel met het vlees. Turbo Vision is dan het skelet waar je het vlees aan kunt hangen. Op die manier kun je flitsende programma's bouwen met een professionele uitstraling.

Het werken met Turbo Vision is geen eenvoudige zaak. De gereedschapskist bevat nogal wat verschillende onderdelen en het vlot werken met het programmeersysteem vereist routine. Maar als je dit eenmaal kunt, ben je in zeer korte tijd in staat prachtig uitziende programma's te schrijven.

Over werken met Turbo Vision is van mijn hand het boek TURBO VISION IN DE PRAKTIJK verschenen. Ook dit boek verscheen bij Uitgeverij Pim Oets. Met de kennis van Turbo Pascal die je inmiddels hebt, moet dit boek nu toegankelijk zijn.

16.6 Opgaven

16.1 Maak de type-definitie van een object dat een Array [1..1000] of Char beheert en daartoe over een procedure Invoer en een procedure Uitvoer beschikt. Het object moet werken volgens het systeem van late binding. Het type object heet TBuffer en moet dynamisch gedeclareerd kunnen worden.

16.2 Maak een erfgenaam van het Buffer-object. Deze erfgenaam heet TMemo en moet ook dynamisch gedeclareerd kunnen worden. Het object TMemo heeft een overschreven procedure Uitvoer.

279

279