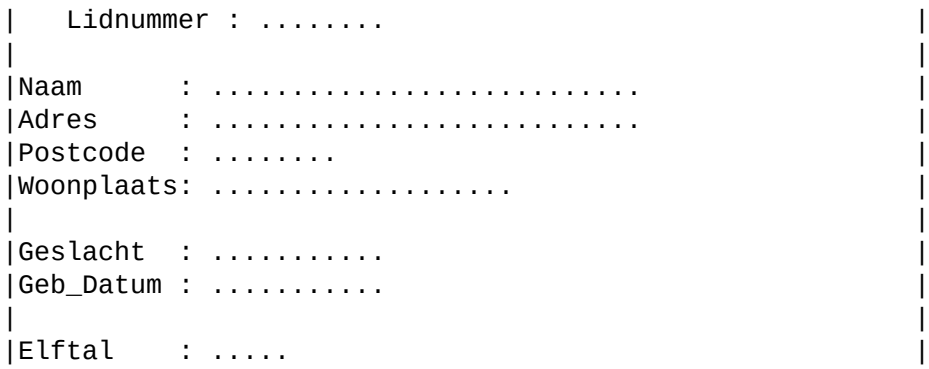
7.1 Het type Record

Een record is een structuur van bij elkaar horende gegevens. Een record kan uit meerdere velden bestaan, die weer van een verschillend type kunnen zijn. Je kunt hierbij denken aan een kaart in een kaartenbak met verschillende gegevens.

Als we een ledenkaart van een voetbalvereniging als voorbeeld nemen, dan zou zo'n kaart er als volgt uit kunnen zien:



Dit zijn de bij elkaar behorende gegevens van een lid van een vereniging. Een record kun je je dus voorstellen als een kaart uit een kaartenbak. In Pascal zouden we de ledenkaart als volgt als een record kunnen definiëren:

```
TYPE
   Ledenkaart = Record
 Lidnummer  : Word;
 Naam       : String[30];
 Adres      : String[30];
 Postcode   : String[7];
 Woonplaats : String[20];
 Geslacht   : Byte;
 Geb_Datum  : String[10];
 Elftal     : Byte;
   END;
```

De overeenkomsten met de ledenkaart zijn duidelijk. Voor elk gegeven is een veld gemaakt, dat overeenstemt met het type gegeven dat in dat veld opgeborgen moet worden. In het programma RECORD_1 wordt de ledenkaart als type gedefinieerd en wordt vervolgens een variabele van dit type gedeclareerd. De kaart

wordt ingevuld en op het beeldscherm getoond:

```
PROGRAM RECORD_1;
USES CRT;

TYPE
   Ledenkaart = Record
      Lidnummer  : Word;
      Naam       : String[30];
      Adres      : String[30];
      Postcode   : String[7];
      Woonplaats : String[20];
      Geslacht   : Byte;
      Geb_Datum  : String[10];
      Elftal     : Byte;
   END;

VAR
   KAART: Ledenkaart;
   Y, X : Byte;

BEGIN
   ClrScr;
   Y := 5;
   X := 20;;
   GotoXY(X,Y);
   Write('Het record Ledenkaart is ',
          SizeOf(Ledenkaart),' bytes groot.');
   Inc(Y);
   FillChar(KAART,SizeOf(KAART),0);
   KAART.Lidnummer  :=  1;
   KAART.Naam       := 'Jansen';
   KAART.Adres      := 'Dorpsstraat 25';
   KAART.PostCode   := '1111 AA';
   KAART.Woonplaats := 'Ons Dorp';
   KAART.Geslacht   :=  0;
   KAART.Geb_Datum  := '15-01-75';
   KAART.Elftal     :=  10;
   WITH KAART DO
   BEGIN
      GotoXY(X,Y);
      Write('Het lidnummer van ');
      IF Geslacht = 0 THEN Write('Meneer ')
      ELSE
      Write('Mevrouw ');
      Write(Naam,' is: ',Lidnummer);
      Inc(Y);
      GotoXY(X,Y);
```

```
      Write('Het adres is: ', Adres);
      Inc(Y);
      GotoXY(X,Y);
      Write(Postcode,' te ',Woonplaats);
      Inc(Y);
      GotoXY(X,Y);;
      CASE Geslacht OF
         0: Write('Hij ');
         1: Write('Zij ')
      END;
      Write('is geboren ',Geb_Datum,
            ' en speelt in het', Elftal,'e elftal');
      Readln
   END
END.
```

Regels:Toelichting:

```
[1]  3-13Definieer het record Ledenkaart.
[2] 14-16Declareer de variabele KAART als type Ledenkaart.
       Declareer X en Y voor het schrijven naar het scherm.
[3]19-21Geef X en Y een beginwaarde en zet de cursor op de
       positie die door X en Y aangewezen wordt.
[4]22-23Schrijf de omvang in bytes van Ledenkaart op het
       scherm.
[5]25       Zet alle velden van KAART op 0.
[6]26-33    Geef alle velden van KAART een waarde.
[7]    34-55     Zet de gegevens van het record KAART op het
       scherm.
```

Toelichting:

[1]Het type Ledenkaart wordt gedefinieerd als type Record. Achter het beschermde woord Record komt geen puntkomma te staan. Direct daarna kunnen de afzonderlijke velden gedeclareerd worden. De definitie van het record wordt afgesloten met END;.

[2]De variabele KAART wordt gedeclareerd als variabele van het type Ledenkaart. De variabelen X en Y worden gebruikt om de cursorpositie op het scherm aan te wijzen.

[3]De variabelen X en Y krijgen beide een beginwaarde en worden samen met GotoXY gebruikt. Telkens als er naar een nieuwe regel wordt gesprongen, wordt Y met 1 verhoogd. Hiervoor gebruiken we de Turbo Pascal-procedure Inc. Deze procedure verhoogt een meegezonden variabele met 1. Zetten we echter Inc(Y,5), dan wordt Y met 5 verhoogd. Telt Inc op, dan heeft Turbo Pascal ook nog een procedure die volgens hetzelfde principe werkt, maar aftrekt. Deze procedure heet Dec.

[4]Vervolgens willen we de omvang van het type Ledenkaart in bytes op het scherm zetten. We gebruiken hiervoor de Turbo Pascal-functie SizeOf. Als je een variabele of een type als parameter aan SizeOf meegeeft, dan wordt het aantal bytes dat dit type of deze variabele groot is geretourneerd.

[5]Turbo Pascal maakt niet automatisch de velden van een gedeclareerd record leeg. We kunnen dit doen door de procedure FillChar te gebruiken. In dit voorbeeld worden alle bytes van de variabele KAART op 0 gezet. Het aantal wordt aangegeven door SizeOF. Nu heb ik de variabele als parameter meegegeven en op regel 23 gaven we het type mee. Voor de uitkomst maakt dit niets uit. Beide zijn voor dit doel te gebruiken.

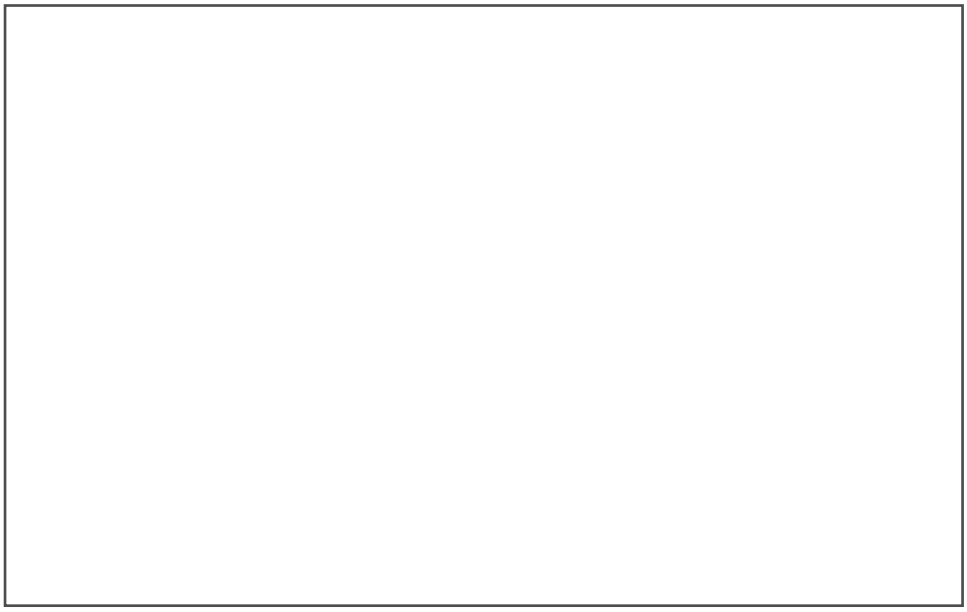
[6]De velden van het record zijn te bereiken door eerst de naam van de variabele te geven, gevolgd door een punt en de

veldnaam. Bijvoorbeeld: KAART.LidNummer.

[7]Een tweede methode is het WITH-statement. Als we schrijven:

```
WITH KAART DO
BEGIN
        ...
END;
```

zijn na het woord BEGIN alle recordvelden te bereiken alsof het gewone variabelen betreft. De END; sluit deze mogelijkheid weer af. In het programma RECORD_1 wordt dit gedemonstreerd. Ook kun je zien hoe je uit gegevens complete zinnen kunt maken met behulp van Write.



Afbeelding 8

Het veld Geslacht is van het type Byte. Deze krijgt de waarde 0 als het een man betreft en de waarde 1 als het een vrouw betreft. Als je in het programma het veld Geslacht op 1 zet, dan krijg je de vrouwelijke variant van de tekst.

Het is niet noodzakelijk om eerst een type te definiëren en vervolgens een variabele te declareren van ditzelfde type. Een record kan ook direct als variabele gedeclareerd worden:

```
VAR
   Kaart     : Record
        Lidnummer : Byte;
```

```
    Naam      : String[30];
    Adres     : String[30];
    ...
END;
```

Hoewel deze manier van declareren uitstekend zal functioneren, kun je het best de eerste methode gebruiken die beter leesbaar is.

7.2 Het variant-record

Het record Ledenkaart dat we zojuist behandeld hebben, is een eenvoudig record met een vaste lijst van velden. Als we leden inschrijven bij een voetbalvereniging, willen we altijd dezelfde gegevens weten. De werkelijkheid van alledag is echter niet altijd zo simpel.

Stel je voor dat je een makelaar bent. Je hebt verschillende soorten woningen in de verkoop, zoals flats, eengezinswoningen en bungalows. Het zal duidelijk zijn dat iemand die geïnteresseerd is in een flat, gedeeltelijk andere informatie wil hebben dan iemand die geïnteresseerd is in een eengezinswoning. Ze zullen beiden willen weten wat de prijs en wat het bouwjaar is. Iemand die een flat koopt, wil ook nog weten hoe hoog het gebouw is, op welke verdieping de flat ligt, wat de maandelijkse servicekosten zijn, enzovoort. Dit soort zaken wil de aspirant-koper van een eengezinswoning helemaal niet weten. Die wil weten of het huis een plat of een schuin dak heeft, of het een hoekwoning is, en hoeveel meter tuin er bij zit.

In dit geval heb je een kaart nodig die zichzelf aanpast aan de soort woning. Turbo Pascal levert zo’n kaart in de vorm van het variant-record.

Het variant-record is een van de moeilijkere onderwerpen in dit boek. Om te kunnen begrijpen hoe een dergelijk record werkt, moeten we eerst even dieper ingaan op de manier waarop een record in het geheugen opgeslagen wordt. Als gegevenseenheid heeft het record een adres in het geheugen. Op dit adres begint het eerste veld van het record. Bij het record Ledenkaart is dit 2 bytes voor het veld Lidnummer. Daarna volgen 31 bytes voor het veld Naam. Bij een string moeten we immers één byte bijtellen voor de lengtebyte. Daarna volgt het veld Adres. De velden staan in volgorde van declaratie. Het record wordt als een soort masker, beginnend op het adres van het record, over dit deel van het geheugen gelegd. Zo zijn we in staat om de afzonderlijke velden

te lezen.

Een variant-record kent naast een vaste lijst van gegevens een variërend deel. Dit variërende deel kan bestaan uit meerdere varianten. Om bij ons voorbeeld van de makelaar te blijven: als het een flat betreft, worden in het variërende deel de gegevens die bij de flat horen opgeslagen. Betreft het een eengezinswoning, dan worden daar de gegevens van een eengezinswoning geschreven. Hetzelfde geldt voor de gegevens van een bungalow.

Een variant-record omvat zoveel bytes als nodig zijn om de vaste lijst van gegevens van de grootste variant in op te bergen. De kleinere varianten, die dezelfde ruimte benutten, gebruiken gewoon een gedeelte van de variabele ruimte niet.

Het programma RECORD_2 laat zien hoe een variant-record werkt:

```
PROGRAM RECORD_2;
USES CRT;

TYPE
   TSoort = (Flat, Eengezin, Bungalow);

Thuis = Record
   Adres      : String[30];
   Bouwjaar   : Word;
   Prijs      : Real;
   Kamers     : Byte;

   CASE Soort : TSoort OF
   Flat       : (Verdieping: Byte;
                 Etages    : Byte;
                 Lift      : Boolean;
                 Balkon    : Boolean;
                 Servicekst: Real);

   Eengezin  : (Hoekwoning : Boolean;
                SchuinDak  : Boolean;
                MtrTuin    : Word);

   Bungalow  : (Vrijstaand : Boolean;
                Schuifpui  : Boolean;
                EigenGrond : Boolean;
                MtrGrond   : Word;
                Badkamers  : Byte)
END;

VAR
   HUIS  : Thuis;
   OMVANG: Word;
BEGIN
   OMVANG := SizeOf(Thuis);
   WITH HUIS DO
   BEGIN
      Adres      := 'Middenweg 124';
      Bouwjaar   := 1978;
      Prijs      := 185000;
      Kamers     := 4;
      Soort      := Eengezin;
      Hoekwoning := False;
      SchuinDak  := True;
      MtrTuin    := 10
   END
```

END.

Regels:Toelichting:

[1]3-4 Definieer het type TSoort.
[2]6-9Definieer de vaste gegevens van het record Thuis.
[3]10Wijs Soort van het type TSoort aan als sleutel voor de vraag welk variant deel gebruikt moet worden.
[4]11-15 Gegevens die bij een flat horen.
[4]16-18 Gegevens die bij een eengezinswoning horen.
[4]19-23Gegevens die bij een bungalow horen.
25-27Declaratie variabelen.
[5]29 Bepaal de omvang van het record.
[6]30-40 Geef de velden die bij een eengezinswoning horen een waarde.

Toelichting:

[1]Het type TSoort is een door de gebruiker gemaakt opsommingstype en heeft drie mogelijke waarden: Flat, Eengezin en Bungalow. Dit opsommingstype wordt gebruikt als sleutel om te kunnen bepalen welke variant van het record gebruikt moet worden.

[2]De vaste gegevens van het record omvatten:

Adres = 31
Bouwjaar = 2
Prijs = 6
Kamers = 1
===
Totaal 40 (bytes)

[3]Hier komen we een oude bekende tegen die op een andere manier gebruikt wordt. In hoofdstuk 5 hebben we CASE OF gebruikt als vereenvoudigde schrijfwijze voor ingewikkelde IF THEN ELSE-constructies. Hier wordt CASE OF gebruikt om te bepalen welke velden in het variante deel van het record geldig zullen zijn. Tegelijkertijd doet deze regel dienst om het veld Soort te declareren als veld van het type TSoort. Soort gebruikt 1 byte in het geheugen. Dit sleutelveld wordt ook wel het "tag-veld" van het record genoemd.

[4] Soort als veld van het type TSoort kan maar drie waarden hebben. Het aantal bytes dat nodig is voor de drie varianten is verschillend:

Flat = 10 bytes
Eengezin = 4 bytes
Bungalow = 6 bytes

Het variante deel van een record moet altijd als laatste deel van het record gedeclareerd worden. In de CASE OF-constructie moet altijd een opsommingstype gebruikt worden. Dat kan dus een eigengemaakt opsommingstype zijn, of een reeds bestaand type als Char, Boolean, Integer, enzovoort. Het maximale aantal varianten is 256.

[5]Als je het programma draait, zul je zien dat Omvang de waarde 51 heeft gekregen. SizeOf geeft dus aan dat het type Thuis 51 bytes groot is. Laten we eens even kijken of dit klopt met het eerdere verhaal:

```
Vaste gegevens = 40
Flat            =   10 (grootste variant)
Soort           =    1
                        ===
Totaal              51 (bytes)
```

Hier is het bewijs dat de verschillende varianten dezelfde ruimte gebruiken.

[6]In het programma krijgen de velden van de vaste lijst en de velden die bij de eengezinswoning horen een waarde. De velden die bij Flat en Bungalow horen zijn wel toegankelijk. Als we die zouden gaan gebruiken terwijl we Eengezin als soort opgeven, wordt het wat rommelig. Veel beginnende programmeurs vragen zich af wat ze hier nu eigenlijk aan hebben. Straks zullen we leren hoe je records in een bestand op schijf opslaat. Met het variant-record is het mogelijk om verschillende soorten gegevens naast elkaar in één bestand op te slaan, met zo min mogelijk ruimteverspilling.

7.3 Het free union-record

Soms is het handig een adres in het geheugen op verschillende manieren te kunnen uitlezen. Daarvoor is het free union-record bedacht. Een free union-record is een variant-record zonder sleutelveld. Het tag-veld ontbreekt. Het eigenaardige is, dat er wel een type voor dit tag-veld wordt opgenomen. Doordat een tag-veld ontbreekt, kunnen alle velden van het record gebruikt worden. Zij overlappen elkaar. Waarschijnlijk maakt een voorbeeld veel meer duidelijk dan een beschrijving van vele zinnen:

```
PROGRAM RECORD_3;
USES CRT;

TYPE
   TFreeUnion = Record
   CASE Integer OF
      1 :(Zin: String[4]);
      2 :(Lengte: Byte;
          Letters: Array[1..4] OF Char);
      3 :(Lengtebyte: Char;
          ASCII: Array[1..4] OF Byte);
   END;

VAR
   FREEUNION: TFreeUnion;
   I: Byte;
   OMVANG: Word;
BEGIN
   FREEUNION.Zin := 'Test';
   OMVANG := SizeOf(FREEUNION);
   ClrScr;
   Writeln
   ('De omvang van FreeUnion is : ',OMVANG,' bytes');
   WITH FREEUNION DO
   BEGIN
      Writeln('In het veld Zin staat: ',Zin);
      Writeln('In het veld Lengte staat: ',Lengte,
              ' dit is het ASCII-teken: ',LengteByte);
      FOR I := 1 TO 4 DO
      Writeln('In Letters ',I,' staat : ',Letters[I],
            '   dit is ASCII-teken nummer ',ASCII[I])
  END;
  Readln
END.
```

Als je het programma uitvoert, kun je duidelijk zien dat de velden van dit record elkaar overlappen:

Afbeelding 9

Regels:Toelichting:

[1]3-11Definitie free union-record.
12-15Declaratie variabelen.
17 Geef beginwaarde aan het veld FREEUNION.Zin.
[2]18Kijk met SizeOf wat de omvang van het record is en plaats dat in OMVANG.
20-21 Schrijf de omvang van het record naar het scherm.
22Gebruik WITH om de velden van het record als gewone variabelen te kunnen lezen.
24 Schrijf het veld FREEUNION.Zin naar het scherm.
[3]25Laat zien wat de lengte van Zin is en welk ASCII-teken daarvoor gebruikt wordt.
[4]27-29Ga een FOR-lus in die vier keer wordt uitgevoerd. Laat telkens zien welk letterteken in het bijbehorende element van Letters staat. Laat tevens zien welk nummer dit letterteken in de ASCII-tabel heeft.

Toelichting:

[1] In de definitie van het record ontbreekt, zoals gezegd, het tag-veld. Er staat:

CASE Integer OF

Dat wil zeggen dat we de diverse alternatieve velden onder een nummer kunnen definiëren. De velden overlappen elkaar. Het veld Lengte heeft hetzelfde adres als Zin[0]. In byte 0 van een string wordt de lengte van de string bijgehouden. Ik heb de lengte als

Byte gedefinieerd, dus kan ik in het veld Lengte de lengte van de string als getal uitlezen. Het veld Lengtebyte heeft ook hetzelfde adres, maar is als type Char gedefinieerd. In dit veld staat nu dus de lengte van de string als letterteken. Dit is letterteken 4 uit de ASCII-tabel.

De array Letters, een array van het type Char, bestaat uit vier elementen van één byte. Elk van deze elementen heeft hetzelfde adres als de vier letterteken die in het veld Zin staan. In Zin[1] staat hetzelfde als in Letters[1].

De array ASCII, een array van het type Byte, wijst ook naar hetzelfde adres. Op die manier kan ik de ASCII-nummers van de verschillende letters uitlezen.

[2]SizeOf(TFreeUnion) vertelt ons dat het record vijf bytes groot is. Ook hier kunnen we dus zien dat de verschillende alternatieven elkaar overlappen.

[3]Door het veld Lengte samen met het veld Lengtebyte naar het scherm te sturen, kunnen we zien wat de lengte van Zin is en welk ASCII-teken hier voor gebruikt wordt.

[4]Binnen de FOR-lus wordt I bij iedere doorgang met 1 verhoogd. Er wordt dus vier keer een element uit de array Letters en uit de array ASCII afgedrukt. In Letters[I] staat het letterteken en in ASCII[I] staat het nummer dat dit letterteken heeft in de ASCII-tabel.

7.4 Opgaven

7.1 Een groothandel heeft een magazijn met een geautomatiseerd beheer. Ieder artikel heeft een record. Declareer een recordtype dat ruimte biedt om de volgende gegevens op te slaan:

ArtikelNummer;
ArtikelNaam;
Prijs;
Leverancier.

ArtikelNummer is een combinatie van twee letters aan het begin, gevolgd door vier cijfers. Een artikelnaam moet passen in maximaal 30 lettertekens en een leverancier wordt herkend aan zijn nummer. Leveranciers worden genummerd van 1 tot 1000.

7.2Natuurlijk zullen niet alle artikelnummers in gebruik zijn. Sommige artikelen komen bijvoorbeeld niet meer in het assortiment voor. In zo'n geval wordt de reden dat deze artikelen niet meer leverbaar zijn vermeld. Deze vermelding wordt opgeslagen in hetzelfde bestand als de artikelen. Het record Artikel moet dus omgedoopt worden naar een variant-record. Als een artikel in gebruik is, moet het gelezen kunnen worden als het record uit opgave 7.1. Als het niet in gebruik is, bestaat het uit:

ArtikelNummer;
ArtikelNaam;
Reden.

Om de reden te kunnen vermelden, moeten er maximaal 50 posities beschikbaar zijn.

7.3De magazijnmeester van de groothandel heeft nu weer andere noten op zijn zang. Het variant-record uit de vorige opgave wil hij niet meer. We vallen dus weer terug op het oorspronkelijke record uit opgave 7.1. Nu wil hij het artikelnummer in zijn onderdelen uit kunnen lezen. De twee beginletters van het artikelnummer duiden een categorie aan. De vier cijfers daarachter geven het nummer binnen de categorie aan. Het record uit opgave 7.1 moet omgezet worden naar een free union-record, waaruit zowel het volledige artikelnummer als de categorie en het nummer van de categorie afzonderlijk te lezen zijn.

107

107

120

120

120