18. De unit STRINGS

18.1 Pascal-strings versus C-formaat

Het oorspronkelijke Pascal bevatte geen apart gegevenstype voor het opslaan van strings. Een string werd gedeclareerd als een rij van lettertekens, dus als: "Array [1..X] of Char". In Turbo Pascal is het type String wel degelijk één van de beschikbare gegevenstypen. Als je "Woord:String[40]" declareert, reserveer je in het geheugen 41 bytes voor de opslag van lettertekens. Er worden 41 bytes gereserveerd, omdat we een extra byte nodig hebben waar Turbo Pascal de lengte van de string in bijhoudt. In één byte past maximaal de waarde 255. Het gevolg hiervan is dat een string in Turbo Pascal maximaal 255 lettertekens kan bevatten.

De programmeertaal C kent deze beperking niet. In C worden strings op een andere manier opgeslagen. Een string onder C heeft geen lengtebyte, maar wordt afgesloten door het teken "0" uit de ASCII-tabel. De met C meegeleverde programmabibliotheek levert de functies om de lengte van de string te meten en om de strings te manipuleren. Aangezien de lengte niet bijgehouden wordt in een lengtebyte, kan een string in C maximaal 65535 lettertekens bevatten.

Omdat programma's steeds vaker met elkaar moesten kunnen communiceren, ontstond er behoefte aan een standaardmethode voor het vastleggen van strings. De standaard is de stringopslag volgens de programmeertaal C geworden.

Om Turbo Pascal up to date te houden, levert Borland sinds de versie 7.0 een aparte unit STRINGS mee. Deze unit stelt ons in staat om in een programma strings te gebruiken volgens het concept van C. Deze strings worden "null terminated strings" genoemd.

18.2 Het type PChar

In de unit STRINGS staat het type PChar gedefinieerd. Het type PChar is een pointer naar een rij lettertekens (Array of Char). PChar gedraagt zich net even anders dan een gewone pointer. Normaal is het zo, dat als je wilt weten waar een pointervariabele naar wijst, je achter de variabelenaam een dakje zet. Een variabele Zin als een pointer naar een string, neem je op als "Zin^".

Bij het type PChar kan het dakje achterwege blijven. PChar is wel een pointer, maar deze pointer laat meteen zien waar hij naar wijst. Je kunt variabelen van het type PChar declareren en daar een waarde aan toewijzen. PChar wijst dan naar een geheugenplaats in het datasegment of naar de stack. Dit is afhankelijk van de vraag of het een globale of een lokale variabele betreft. Op de plaats waar de variabele naar wijst, staat de string die je hebt toegewezen.

De zaak loopt fout als je de string groter maakt. In dat geval wordt dat wat achter de string stond (en dat kan van alles zijn) zonder meer overschreven.

Het programma STRING_1 laat zien hoe dit in de praktijk werkt:

```
PROGRAM STRING_1;
USES CRT, STRINGS;
VAR
   ZIN, PRIJS: PChar;
BEGIN
   ClrScr;
   ZIN   := 'Dit kost: ';
   PRIJS := 'fl.1000,--';
   Writeln(ZIN,PRIJS);
   StrCat(ZIN,'niets ');
   Writeln(ZIN,PRIJS);
   ReadKey
END.
```

Dit stukje programma voert twee maal een Writeln-statement uit met de velden ZIN en PRIJS. De eerste keer verschijnt op scherm: "Dit kost : fl.1000,--". Zodra de eerste Writeln-opdracht is afgerond, wordt de procedure StrCat uit de unit STRINGS uitgevoerd en wordt aan het veld ZIN "niets " toegevoegd. Als nu de tweede Writeln-opdracht uitgevoerd wordt, blijkt de inhoud van het veld PRIJS te zijn overschreven. Oorspronkelijk stonden de twee velden als volgt in het geheugen:

D i t k o s t : #0 f l . 1 0 0 0 , - - #0

Na de overschrijving staat er:

D i t k o s t : n i e t s #0 0 0 , - - #0

De tweede schrijfopdracht leidt tot de verwarrende boodschap:

Dit kost niets iets

Omdat het veld ZIN is uitgebreid, is het gegevensgebied van het veld PRIJS overschreven. Op de plaats waar eerst de "f" stond van "fl.1000,--", staat nu de "i" uit het woordje "niets". Het teken #0 is door de uitbreiding van ZIN een zestal plaatsen naar rechts geschoven en sluit nu zowel de gegevens van het veld

ZIN als van het veld PRIJS af. Daarom is de inhoud van het veld PRIJS nu veranderd van "fl.1000,--" in "iets ".

18.3 PChar dynamisch declareren

Het is vervelend als de inhoud van velden ongewenst verandert. Toch heeft de opslag van strings met een type PChar grote voordelen. PChar verspilt namelijk geen enkele byte. Als je een Turbo Pascal-string declareert, declareer je die als een string van een bepaalde lengte. Als je de lengte-aanduiding weglaat, reserveert Turbo Pascal 256 bytes. Ook al is het gegeven dat je wilt opslaan maar één byte groot, er worden 256 bytes gereserveerd. Bij PChar is dat niet zo. De gegevens liggen naast elkaar in het geheugen, van elkaar gescheiden door het teken #0.

Om te voorkomen dat gegevens ongewild overschreven worden doordat strings groter worden, of doordat er gaten tussen de opgeslagen gegevens vallen omdat strings kleiner worden, kun je variabelen dynamisch declareren. Je moet dan ruimte op de heap reserveren en daar de gegevens in plaatsen. Als je nu iets wilt veranderen, dan declareer je gewoon nieuwe ruimte, je zet daar de gewijzigde string in, en je geeft de oude ruimte weer vrij. De unit STRINGS levert een aantal procedures om ruimte te reserveren en vrij te geven. Het volgende programma laat hiervan een voorbeeld zien:

```
PROGRAM STRING_2;
USES CRT, STRINGS;
VAR
   ZIN, PRIJS: PChar;
   FUNCTION VulAan(VAR S:PChar;Aanvulling:PChar): PChar;
   VAR
      NIEUWE_STR: PChar;
   BEGIN
      GetMem(NIEUWE_STR,StrLen(S)+StrLen(Aanvulling));
      StrCopy(NIEUWE_STR,S);
      StrCat(NIEUWE_STR,Aanvulling);
      StrDispose(S);
      VulAan := NIEUWE_STR
   END;

BEGIN
   ClrScr;
   ZIN   := StrNew('Dit kost: ');
   PRIJS := StrNew('fl.1000.--');
   Writeln(ZIN,PRIJS);
   ZIN   := VulAan(ZIN,' niets');
   Writeln(ZIN,PRIJS);
   ReadKey;
   StrDispose(ZIN);
   StrDispose(PRIJS)
END.
```

Regels: Toelichting:

```
2         Gebruik de units CRT en STRINGS.
3-4Declareer de benodigde globale variabelen.
```

[2]5-14 Function VulAan(VAR S:PChar;Aanvulling:PChar):PChar.

[3]6-7Declareer een lokale variabele van het type PChar om als nieuwe string te dienen.

```
   [4]9         Reserveer geheugen voor de variabelen S en
Aanvulling.
   [5]10       Kopieer de parameter S naar het veld NIEUWE_STR.
   [6]11       Voeg de parameter Aanvulling toe aan NIEUWE_STR.
   [7]12       Geef het geheugen dat gereserveerd was voor S weer
vrij.
13         Geef de functie VulAan de waarde in NIEUWE_STR.
```

15-25Hoofdprogramma.

[1]17-18Declareer twee variabelen van het type PChar op de heap en geef ze een waarde.

```
19         Toon de waarden in ZIN en PRIJS.
   [2]20       Roep de functie VulAan aan om tekst aan ZIN toe te
voegen.
   [8]21       Toon opnieuw de waarden in ZIN en PRIJS.
   [9]23-24Geef het geheugen voor ZIN en PRIJS vrij.
```

Toelichting:

[1]De STRINGS-functie StrNew reserveert de hoeveelheid geheugenruimte op de heap die nodig is om de meegekregen tekst in op te bergen. Dit is altijd het aantal lettertekens van de tekst plus 1. In de extra byte wordt dan het afsluitteken #0 gezet.

[2]De unit STRINGS bevat een functie StrCat om twee strings aan elkaar te plakken. StrCat knoopt eenvoudigweg twee strings aan elkaar en overschrijft datgene wat er oorspronkelijk achter de uit te breiden string stond. Dit willen we natuurlijk niet. Vandaar dat we hier de functie VulAan gebruiken. Deze functie voegt twee strings samen en retourneert een pointer naar de samengevoegde tekst.

[3]In de functie VulAan wordt een lokale variabele van het type PChar gedeclareerd.

[4]GetMem reserveert voldoende geheugen op de heap om de tekst in de parameters S en Aanvulling op te bergen. Om het aantal bytes te bepalen, wordt de functie StrLen gebruikt. Deze functie retourneert de lengte van de string en is vergelijkbaar met de Turbo Pascal-functie Length.

[5]De STRINGS-functie StrCopy(NIEUWE_STR, S) kopieert de inhoud van S naar het veld NIEUWE_STR.

[6]StrCat(NIEUWE_STR, Aanvulling) kopieert de inhoud van de

parameter Aanvulling naar de variabele NIEUWE_STR en koppelt het aan de reeds bestaande inhoud. We beschikken in Turbo Pascal ook nog over een functie StrLCat. Deze functie werkt volgens hetzelfde principe als StrCat, maar bij StrLCat kan tevens opgegeven worden hoeveel bytes er maximaal gekopieerd mogen worden.

[7]De geheugenruimte voor ZIN kan nu met de STRINGS-procedure StrDispose vrijgegeven worden. De functie VulAan krijgt de waarde die in de variabele NIEUWE_STR staat. VulAan retourneert dus een pointer naar de samengevoegde tekst.

[8]Als we, teruggekeerd in het hoofdprogramma, de inhoud van ZIN en PRIJS bekijken, zien we dat het veld PRIJS nu niet overschreven is. Dit hebben we voorkomen door een nieuw veld ZIN te maken en het oude veld te vernietigen. Het nieuwe veld heeft de hoeveelheid geheugen gekregen die nodig is om de samengevoegde tekst in op te bergen. Op deze manier kan er nooit overschreven worden.

[9]Als je de inhoud van de velden niet meer nodig hebt, kan de geheugenruimte worden vrijgegeven.

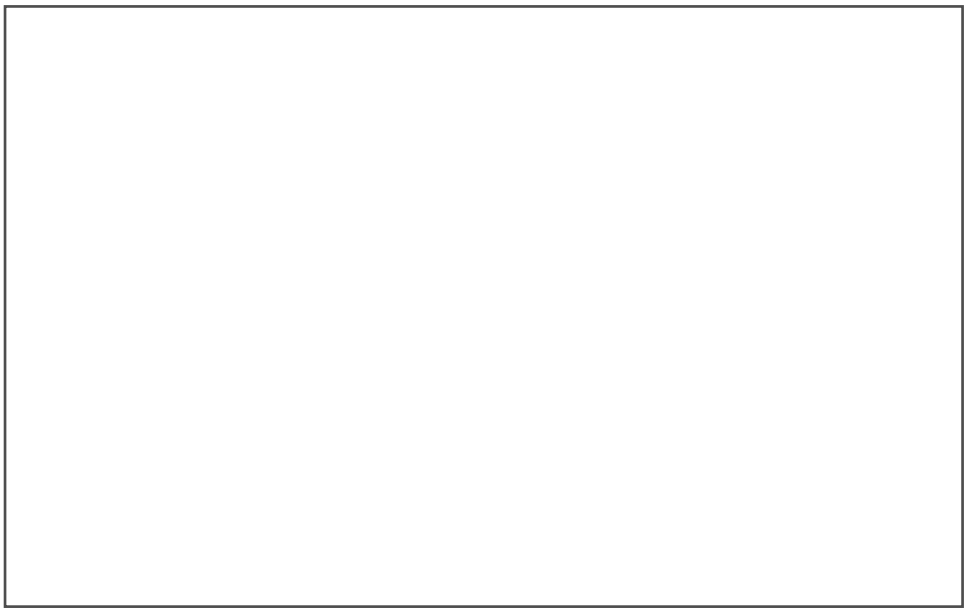
De volgende paragraaf bevat een demonstratie van de verschillende functies van de unit STRINGS.

18.4 STRINGS-functies gebruiken

Het programma STRING_3 demonstreert het gebruik van STRINGS-functies:

```
PROGRAM STRING_3;
USES CRT, STRINGS;
VAR
   ZIN, ZIN_2: PChar;
   PZIN      : String;
   P         : Pointer;

   FUNCTION StrInsert
   (VAR S:PChar;Aanvulling:PChar;Positie:Word):PChar;
   VAR
      NIEUW: PChar;

   BEGIN
      GetMem(NIEUW,StrLen(S)+StrLen(Aanvulling)+1);
      StrLCopy(NIEUW,S,Positie);
      StrCat(NIEUW,Aanvulling);
      StrMove(StrEnd(NIEUW),PChar(@S[Positie]),
              StrLen(@S[Positie]));
      StrDispose(S);
      StrInsert := NIEUW
   END;

   FUNCTION StrDel(VAR S:PChar;Van,Tot:Word):PChar;
   VAR
      NIEUW: PChar;

   BEGIN
      GetMem(NIEUW,StrLen(S)-(Tot-Van)+1);
      StrLCopy(NIEUW,S,Van);
      StrCopy(StrEnd(NIEUW),PChar(@S[Tot]));
      StrDispose(S);
      StrDel := NIEUW;
   END;

BEGIN
   ClrScr;
   ZIN := StrNew('Turbo Pascal kent het type String');
   ZIN_2 := StrNew
          ('Turbo Pascal heeft ook het type PChar');
   Writeln('ZIN         : ',ZIN);
   Writeln('ZIN 2       : ',ZIN_2);
   Writeln('StrLen      : ',StrLen(ZIN),' Bytes');
   P := Pointer(StrEnd(ZIN));
   Writeln('StrEnd      : ',Seg(P^),':',Ofs(P^));
   ZIN := StrInsert
        (ZIN,' voorzien van lengte byte ',StrLen(ZIN));
```

```
   Writeln('StrInsert   : ',ZIN);
   ZIN_2 := StrDel(ZIN_2,18,22);
   Writeln('StrDel      : ',ZIN_2);
   CASE StrComp(ZIN,ZIN_2) OF -(MaxInt+1) .. -1  :
      Writeln('StrComp     : ZIN is kleiner dan ZIN_2');
      0:
      Writeln('StrComp     : ZIN is gelijk aan ZIN_2');
      1 .. Maxint:
      Writeln('StrComp     : ZIN is groter dan ZIN_2')
   END;
   IF StrPos(ZIN,'voorzien') <> NIL THEN
   Writeln('StrPos      : ',StrPos(ZIN,'voorzien'));
   PZIN := StrPas(StrRScan(ZIN_2,'t'));
   Writeln('StrPas      : ',PZIN);
   Writeln('StrUpper    : ',StrUpper(ZIN));
   Writeln('StrLower    : ',StrLower(ZIN_2));
   ReadKey;
   StrDispose(ZIN);
   StrDispose(ZIN_2)
END.
```



Afbeelding 17

Regels: Toelichting:

[1]3-6 Declareer de benodigde globale variabelen.

[5]7-19 Function StrInsert(VAR S:PChar;Aanvulling:
PChar;Positie:Word):PChar.

9-10 Declareer een lokale variabele van het type PChar.

[6]12 Reserveer geheugen op de heap.

[7]13 Zet de inhoud van de parameter S in het veld NIEUW.

[8]14 Kopieer de inhoud van de parameter Aanvulling in het veld NIEUW.

[9]15-16 Kopieer het restant van de parameter S in het veld NIEUW.

[10]17 Geef het geheugen van de VAR-parameter S vrij.

[11]18 Retourneer een PChar die naar het veld NIEUW wijst.

[12]20-29Function StrDel
(VAR S:PChar;Aanvulling:PChar;Van,Tot:Word):PChar.

21-22Declareer een lokale variabele van het type PChar.

[13]24 Reserveer geheugen voor de nieuwe string.

[14]25 Kopieer de inhoud van de parameter S, tot de positie aangegeven door de parameter Van, in het veld NIEUW.

[15]26 Koppel de inhoud van de Parameter S, vanaf de positie aangegeven door de parameter Tot, aan het einde van het veld NIEUW.

```
27        Geef het geheugen van de VAR-parameter S vrij.
28        Retourneer een type PChar die naar NIEUW wijst.
```

30-62 Hoofdprogramma.

[2]32-36Zet een waarde in de velden ZIN en ZIN_2 en toon de waarden op het scherm.

```
[3]37       Toon de lengte van het veld ZIN in bytes.
[4]38-39    Toon het adres van de afsluitende #0.
[5]40-42Voeg in ZIN de doorgegeven tekst in en stuur de nieuwe
       ZIN naar het scherm.
[12]43-44Verwijder de tekst tussen de doorgegeven posities.
[18]45-52   Vergelijk ZIN met ZIN_2 en geef het resultaat
       door.
[19]53-54   Toon de inhoud van ZIN vanaf het woord "voorzien".
[20]55-56Maak van de inhoud van ZIN_2, vanaf de laatste letter
       "t" die voorkomt, een Turbo Pascal-string.
[21]57      Stuur ZIN in hoofdletters naar het scherm.
[22]58      Stuur ZIN_2 in kleine letters naar het scherm.
[23]60-61   Geef de geheugenruimte voor ZIN en ZIN_2 vrij.
```

Toelichting:

[1]De globale variabelen ZIN en ZIN_2 zijn van het type PChar en worden gebruikt om de mogelijkheden van de unit STRINGS te demonstreren. PZIN is een Turbo Pascal-string. Deze string wordt gebruikt om het type PChar naar het type String te converteren. De pointervariabele wordt ten slotte gebruikt om een adres uit te lezen.

[2]De STRINGS-procedure StrNew reserveert ruimte op de heap. StrNew declareert hiervoor het aantal lettertekens in de tekst plus 1. In de byte die wordt toegevoegd, wordt het afsluitingsteken #0 gezet. StrNew retourneert een pointer van het type PChar.

[3]De functie StrLen retourneert de lengte van de tekst waar een type PChar naar wijst. De lengte wordt door StrLen berekend, exclusief het teken #0.

[4]StrEnd retourneert een pointer van het type PChar. De door StrEnd geretourneerde pointer wijst naar het einde van de string die aan StrEnd werd doorgegeven. In dit geval retourneert StrEnd dus een pointer naar het teken #0 dat ZIN afsluit.

[5]In de unit STRINGS ontbreekt een functie om een stukje tekst in te kunnen voegen in een string. Deze functie hebben we nu zelf gemaakt. De functie StrInsert voegt op de aangegeven positie een stukje tekst in en retourneert een PChar die naar de nieuwe tekst wijst.

[6]Om dit te kunnen doen, wordt eerst voldoende geheugen gereserveerd om zowel het veld ZIN als het veld Aanvulling in op te bergen. Er moet één byte extra gereserveerd worden voor het teken #0.

[7]StrLCopy kopieert het aantal bytes dat aangegeven staat in de parameter Positie, van de parameter S naar het veld NIEUW. Nu bevat NIEUW de tekst vanaf het begin tot aan de waarde in de parameter Van. StrLCopy retourneert een PChar naar het veld NIEUW.

[8]StrCat kopieert de tweede parameter naar het einde van de eerste. Met andere woorden: StrCat voegt de twee parameters samen tot één nieuwe string. StrCat retourneert een PChar naar het begin van de eerste parameter. Nu is dus de aanvulling toegevoegd aan de tekst die al in het veld NIEUW stond.

[9]StrMove verplaatst een aantal tekens uit een string naar een andere string en retourneert een PChar die wijst naar de bestemming. StrMove krijgt twee parameters mee. De eerste is de bestemming. In het programma is dit het veld NIEUW. Omdat de tekst toegevoegd moet worden aan de reeds bestaande tekst, en de bestaande tekst niet overschreven mag worden, wordt hier StrEnd(NIEUW) doorgegeven. StrEnd retourneert een PChar die wijst naar het einde van de string NIEUW.

Als tweede parameter geven we het begin van de tekst door die rechts staat van de plaats, aangegeven door het veld Positie. Het type PChar wijst naar een array van het type Char. S[Positie] geeft het teken weer dat op de plaats staat, aangegeven door Positie. @S[Positie] geeft het adres van S[Positie].

De typecasting PChar(@S[Positie]) geeft het adres door van S[Positie] in de vorm van een type PChar aan StrMove. Dezelfde typecasting kunnen we toepassen om de lengte van de resterende string te achterhalen. Hiertoe wordt StrLen aangeroepen. Als hier meer bytes worden doorgegeven dan de te verplaatsen tekst groot is, zet de functie StrMove hier het teken #0 voor. Dit heeft dan tot gevolg dat de string onmiddellijk na de te verplaatsen tekst wordt afgebroken. Met de aanroep van StrMove wordt het stukje tekst rechts van het veld Positie in het veld NIEUW geplaatst. De tekst in de parameter Aanvulling is op de aangegeven plaats in de inhoud van S ingevoegd. StrMove doet hier precies hetzelfde als StrCat.

[10]StrDipose doet het omgekeerde van StrNew. Hier wordt het aantal bytes op de heap vrijgegeven, dat bepaald wordt door de lengte van de tekst plus 1.

[11]Ten slotte krijgt de functie StrInsert de waarde die in het veld NIEUW staat.

[12]Een functie die ook ontbreekt in de unit STRINGS is de mogelijkheid om een substring uit een string te verwijderen. De functie StrDel in het programma STRING_3 voorziet hierin.

[13]Hier wordt eerst de hoeveelheid geheugen gereserveerd die nodig is om de oorspronkelijke tekst minus de te verwijderen tekst in op te slaan.

[14]StrLCopy kopieert het aantal bytes dat aangegeven wordt door de parameter Van naar het veld NIEUW.

[15]StrCopy kopieert de tweede parameter naar de eerste en

retourneert een pointer naar de bestemming. Omdat hier de bestemming het einde van het veld NIEUW is, wordt als parameter StrEnd(NIEUW) doorgegeven. De tekst die toegevoegd wordt, begint bij het veld Tot. De typecasting PChar(@S[Tot]) geeft een parameter door van het type PChar. De tekst hiervan begint bij het teken dat aangegeven wordt door de parameter Tot. Hier had ook StrCat gebruikt kunnen worden. Naast StrCopy en StrLCopy heeft de unit STRINGS nog een kopieerfunctie beschikbaar. Dit is de functie StrECopy. Deze functie werkt volgens hetzelfde principe als StrCopy en retourneert een pointer naar het einde van de bestemmingsstring.

[16]Het geheugen voor de parameter S wordt met DisposeStr vrijgegeven.

[17]StrDel retourneert een PChar die wijst naar de tekst waaruit zojuist een stukje verwijderd is.

[18]De STRINGS-functie StrComp vergelijkt twee strings. Hiertoe wordt de inhoud van de beide strings byte voor byte met elkaar vergeleken. Zodra StrComp twee tekens vindt die van elkaar verschillen, wordt de vergelijking onderbroken en wordt het verschil in waarde tussen de twee tekens geretourneerd. Als in deze vergelijking het teken in de eerste parameter een hogere waarde heeft dan het teken in de tweede parameter, dan wordt het verschil positief geretourneerd. Als het een lagere waarde heeft, dan wordt het verschil negatief teruggegeven. Als geen verschil gevonden wordt, dan wordt er een 0 geretourneerd. Als één van de strings geen bytes meer bevat, wordt de vergelijking onderbroken en wordt een 0 geretourneerd.

Voor StrComp is in een vergelijking de letter "A" kleiner dan de letter "a". De unit STRINGS voorziet met StrIComp in een methode waarbij in een vergelijking beide letters dezelfde waarde hebben. Met andere woorden: StrIComp maakt geen onderscheid tussen hoofdletters en kleine letters. StrIComp werkt verder op dezelfde manier als StrComp.

Of het nog niet genoeg is, biedt de unit STRINGS ook nog de functies StrLComp en StrLiComp. Deze twee vergelijkingsmethoden werken volgens hetzelfde principe als StrComp en StrIComp. Alleen wordt hier een derde parameter meegegeven met het maximale aantal bytes dat vergeleken mag worden.

Met het aanroepen van StrComp(ZIN,ZIN_2) worden de strings met elkaar vergeleken. De CASE OF-constructie reageert vervolgens op de waarde. In de constructie wordt de voorgedefinieerde Turbo Pascal-constante MaxInt gebruikt. Deze constante geeft het aantal bytes aan dat maximaal in een type Integer past. Positief is dit het aantal 32767, negatief mag daar 1 bij worden opgeteld.

[19]De functie StrPos zoekt naar een substring in een string. Als deze gevonden is, wordt een PChar geretourneerd die naar de gezochte substring wijst. Wordt de substring niet gevonden, dan retourneert StrPos een NIL. Vandaar dat in het voorbeeld eerst onderzocht wordt of StrPos een NIL retourneert. Als dit niet het geval is, dan wordt het resultaat van het aanroepen van StrPos naar het scherm gezonden. StrPos retourneert een PChar, die wijst naar het begin van de gezochte substring.

[20]Uiteraard moeten we een string die opgeslagen is als PChar, weer kunnen veranderen in een gewone Turbo Pascal-string. Hier zorgt de functie StrPas voor. Omdat we het gedeelte uit ZIN_2 dat bij de laatste letter

"t" begint wilden kopiëren, gebruiken we hier de STRINGS-functie StrRScan. Deze functie doorzoekt de string van rechts naar links en retourneert een type PChar die wijst naar de gevonden letter. Omdat StrRScan van rechts naar links zoekt, krijg je het adres van de letter waar deze voor de laatste keer in de tekst voorkomt. StrScan werkt volgens hetzelfde principe als StrRScan, maar StrScan zoekt van links naar rechts en wijst naar het adres van de letter waar deze voor de eerste keer in de tekst voorkomt. Als de letter niet gevonden wordt, dan wordt een NIL geretourneerd. De Turbo Pascal-string bevat met de aanroep van StrPas de tekst rechts van de laatste letter "t" in de tekst.

De functie die het omgekeerde doet van StrPas wordt ook door de unit STRINGS geleverd. De functie StrPCopy kopieert een Turbo Pascal-string naar een string van het type PChar en retourneert een PChar die wijst naar de bestemming.

[21]De functie StrUpper verandert alle letters in de doorgezonden parameter in hoofdletters.

[22]De functie StrLower maakt ten slotte van alle hoofdletters kleine letters.

[23]Na een korte pauze om het scherm te kunnen lezen, wordt het geheugen voor ZIN en ZIN_2 vrijgegeven en wordt het programma beëindigd.

```
--------------------
377

          377

          377
```