19. Compileren

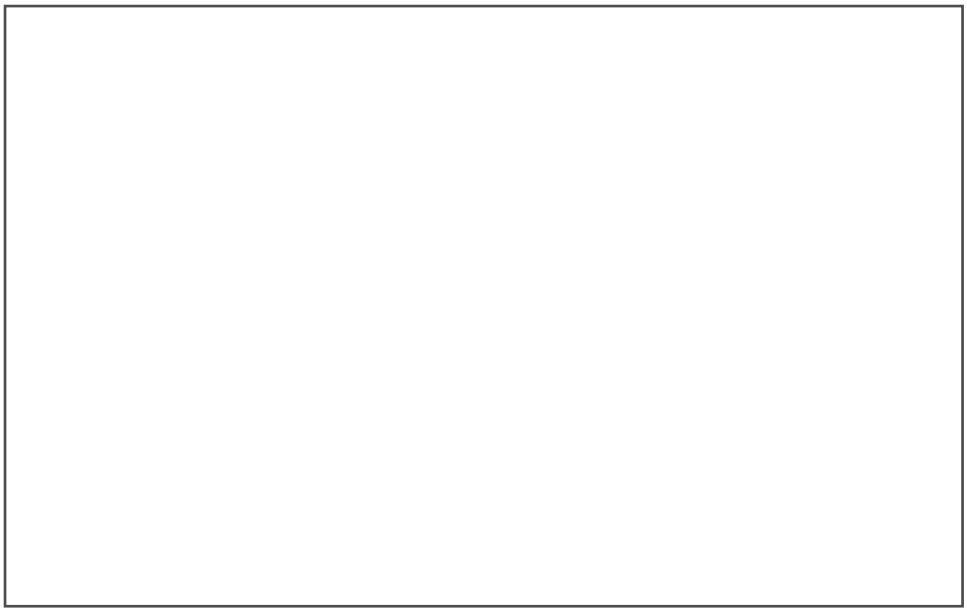
19.1 Compileeropties

In de voorgaande hoofdstukken is een aantal keren gezegd: "aan het begin van het programma moet de optie {$X+} gezet worden". Inhoudelijk zijn de verschillende compileeropties echter nog niet behandeld. In dit hoofdstuk zullen we die onder de loep nemen.

In de eerste plaats moet het duidelijk zijn dat je met compileeropties de compiler kunt instellen om een uitvoerbaar programma te maken in de vorm die je wenst. Compileeropties kunnen overal in een programma opgenomen worden. Opties die voor een heel programma of unit gelden, worden over het algemeen op de allereerste regel gezet, nog vóór het beschermde woord PROGRAM of UNIT. Een optie wordt tussen accolades geplaatst. Onmiddellijk na het plaatsen van de linker accolade wordt een dollarteken gezet, gevolgd door de gewenste optie. Als er een aantal opties ingesteld moet worden, dan worden die met een komma van elkaar gescheiden. Bij volgende opties in dezelfde regel blijven dollartekens achterwege. De vorm is:

{$O,D+}

Een andere mogelijkheid is om de opties in te stellen via de werkomgeving van Turbo Pascal. In dit hoofdstuk worden de verschillende opties inhoudelijk behandeld en wordt tevens aangeven op welke manier de opties aan het begin van een programma of een unit geplaatst kunnen worden. De menukeuzen «*Options*» en «*Compiler*» laten een invoerscherm zien als in afbeelding 18.



Afbeelding 18

19.2 Opties voor het aanmaken van code

Force far calls

Deze optie komt overeen met {$F+} en {$F-}. Het plusje achter de optie betekent: "de optie staat aan". Een minnetje erachter betekent: "de optie staat uit". Als in de werkomgeving (zie afbeelding 18) de optie aangezet wordt, verschijnt er een X tussen de vierkante haken. De opties kunnen hier aan- of uitgezet worden door middel van het klikken met de muis of door de cursor op het veld te plaatsen en op de spatiebalk te drukken.

Als vanuit een procedure of functie een andere procedure of functie aangeroepen wordt, dan wordt het terugkeeradres op de stack gezet. Dit is uitgebreid behandeld in hoofdstuk 6. In hoofdstuk 12 waar alle geheugenperikelen aan de orde kwamen, zagen we hoe een adres in de vorm van een segment en een offset is opgebouwd.

De instructies voor de microprocessor bevinden zich in het codesegment. De compiler kent twee modellen: het Near- en het Far-model. In het Near-model zetten alle procedures en functies die alleen in het implementatie-gedeelte van de unit zijn gedeclareerd, slechts twee bytes op de stack om het terugkeeradres aan te geven. Dit kan omdat in zo'n geval het terugkeeradres altijd binnen hetzelfde codesegment ligt. Alleen de offset van het adres is dan voldoende. In het Far-model wordt er van uitgegaan dat een aan te roepen procedure of functie buiten het geldende codesegment ligt. In zo'n geval is een volledig adres nodig om terug te kunnen keren. Er zijn dan vier bytes nodig om het volledige adres op de stack te zetten. Het volledige adres is dan zowel een aanduiding van het segment als van de offset. Als procedures of functies in het interface-gedeelte gedeclareerd zijn, worden ze altijd, ongeacht de stand van de $F-optie, volgens het Far-model gecompileerd. Als de optie {$F+} niet gezet is, neemt de compiler automatisch de optie {$F-} aan.

Overlays allowed

Deze optie komt overeen met de opties {$O+} en {$O-}. Het begrip overlays is inhoudelijk behandeld in hoofdstuk 10, "Units en Overlays". Als er geen $O-optie gezet is, neemt de compiler automatisch de optie {$O-} aan. Als de optie {$O+} gezet wordt, betekent dit niet dat het verplicht is de unit als overlay in je programma op te nemen. De compiler neemt echter wel een aantal voorzorgsmaatregelen om het mogelijk te maken de unit als overlay te laten functioneren. Als een unit als overlay gebruikt wordt, mag er zich niets bevinden in het initiërings-gedeelte van een unit tussen de laatste BEGIN en END. Als hier zaken staan die bij de start van het programma geïnitieerd moeten worden, kan de unit niet als overlay functioneren. Je moet er dan van afzien om van de unit een overlay te maken, of je moet de zaken op een andere manier initiëren.

Word align data

De met deze optie overeenstemmende aanduidingen zijn {$A+} en {$A-}. Als deze optie "aan" staat, worden gegevens, als het geen lettertekens zijn, op even adressen gezet. Als de optie "uit" staat, worden de gegevens op de geheugenplaats gezet die aan de beurt is, ongeacht of dit een even of een oneven adres is. Als de gegevens alleen op even adressen staan, neemt de snelheid van het programma toe. Je hebt echter wel meer geheugen nodig. Als je dus erg veel gegevens in het geheugen moet bewaren, moet je de optie uitzetten. Het programma wordt dan wel langzamer, maar je beschikt over meer geheugen.

286 instructions

De keuze voor 286 instructions betreft de opties {$G+} en {$G-}. De Turbo Pascal-compiler maakt code aan volgens de opdrachtenset van de 8086- en de 8088-microprocessor die een geheugen van één megabyte kunnen adresseren. De 80286-microprocessor en hoger kunnen veel meer geheugen aan. De 80286-microprocessor beschikt ook over een grotere opdrachtenset. Als de optie {$G+} gezet is, maakt de compiler code aan volgens de uitgebreide 80286-opdrachtenset.

Programma’s die met deze optie gecompileerd zijn, kunnen niet draaien op een computer met een 8086- of 8088-microprocessor. Als een programma kan werken met meer dan één megabyte geheugen, wordt dat de "protected mode" genoemd. De Borland Pascal-compiler kan dit soort programma’s maken. In de protected mode neemt de compiler aan dat, als een $G-optie ontbreekt, de optie {$G+} bedoeld wordt. Een programma dat werkt binnen de grenzen van één megabyte wordt een programma in de "real mode" genoemd. De Turbo Pascal-compiler levert programma’s in de real mode. Deze compiler neemt standaard de optie {$G-} aan.

19.3 Foutmeldingopties

Range checking

Deze optie controleert of bij het gebruik van arrays en strings de grenzen van het aangegeven bereik niet worden overschreden. De optie-aanduidingen zijn: {$R+} en {$R-}. Als er meer waarden in een array gestopt worden dan volgens de capaciteit van de array mogelijk is, wordt de uitvoering van het programma met een foutmelding onderbroken. Als er geen controle op het bereik plaatsvindt, dan kunnen er zaken in het geheugen worden overschreven. Dit heeft vaak desastreuze gevolgen. Als er geen $R-optie wordt aangegeven, neemt de compiler aan dat er geen bereikcontrole gewenst is. De compiler werkt dan alsof de

optie {$R-} gezet is.

Tijdens de ontwikkeling van een programma is het verstandig de optie {$R+} op te nemen. Fouten die optreden door niet-bedoelde overschrijving van het geheugen, zijn moeilijk op te sporen. Met deze optie "aan" word je in ieder geval gewaarschuwd. Als het programma klaar is en in de praktijk gebruikt gaat worden, dan wordt de optie meestal afgezet. De snelheid van het programma neemt dan toe.

Stack checking

Deze optie controleert bij het aanroepen van een procedure of functie of er voldoende ruimte op de stack is om de lokale variabelen in te plaatsen. Deze optie wordt aangeduid met {$S+} en {$S-}. Als de optie "aan" staat, wordt de uitvoering van het programma onderbroken als er te weinig ruimte op de stack aanwezig is. De grootte van de stack is instelbaar (zie hiervoor paragraaf 19.8).

I/O checking

Deze controle kan worden gemanipuleerd met de opties {$I+} en {$I-}. Als de optie "aan" staat, wordt de uitvoering van een programma onderbroken met een foutmelding als er een in- of uitvoerfout optreedt. Standaard staat de optie "aan". Als de optie "uit" staat, kun je natuurlijk altijd met de Turbo Pascal-variabele IOResult controleren of een in- of uitvoerbewerking foutloos verlopen is.

Overflow checking

Deze optie waakt ervoor dat rekenkundige bewerkingen van gehele getallen niet met grotere waarden worden uitgevoerd dan de operatoren kunnen verwerken. Er treedt dan een zogenaamde overflow op. De optie wordt aangegeven met {$Q+} en {$Q-}. Als de optie {$Q+} gezet is, wordt de uitvoering van het programma, zodra zich een overflow voordoet, onderbroken door een foutmelding.

De operatoren waar de optie over waakt zijn: +, -, *, Abs, Sqr, Succ, AND en Pred. Inc en Dec vallen buiten de werking van deze optie. Indien de $Q-optie niet gezet is, werkt de compiler alsof {$Q-} gezet is. In hoofdstuk 13 "Expressies en operatoren" worden de capaciteit en de werking toegelicht.

19.4 Optie-vormen

Strict VAR-strings

Deze optie, aangeduid met {$V+} en {$V-}, voert een controle uit op het gebruik van strings als VAR-parameters. Als de optie "aan" staat, controleert de compiler of de aan een procedure of functie doorgegeven string van hetzelfde type is als het type in de parameterlijst van de procedure of functie. Is dit niet het geval, dan wordt het compileren onderbroken met een foutmelding van de compiler. Als {$V-}

gezet is, blijft een dergelijke controle achterwege. Standaard neemt de compiler aan dat {$V+} gezet is.

Complete boolean-evaluatie

In een booleaanse expressie is het soms al na het verwerken van een deel van de expressie duidelijk of de hele expressie waar of niet waar is. Als we zeggen:

IF (A=1) OR (B=2)

is de expressie al waar als A de waarde 1 heeft. Het is dan eigenlijk niet meer nodig om ook het tweede deel van de expressie te evalueren. Hetzelfde geldt voor AND-expressies. Bijvoorbeeld:

IF (A=1) AND (B=2)

Als A een andere waarde heeft dan 1, is de expressie al niet waar en hoeft B=2 niet meer geëvalueerd te worden. Deze optie die gezet wordt met {$B+} en {$B-}, maakt een gedeeltelijke evaluatie als de optie {$B+} gezet is. Als de optie niet gezet is, werkt de compiler alsof {$B-} gezet is.

Extended syntax

De Extended syntax-optie, aangegeven met {$X+} en {$X-}, maakt het mogelijk om door de programmeur gemaakte functies te gebruiken alsof het procedures zijn. Als de optie "aan" staat, is het niet noodzakelijk om iets te doen met de waarde die de functie retourneert. Deze optie moet tevens "aan" gezet worden als er gewerkt wordt met strings die afgesloten worden met een #0. Als je gebruik maakt van het type PChar, moet de optie {$X+} altijd gezet zijn. Standaard wordt deze ook gezet.

Typed @-operator

Deze optie wordt geschreven met {$T+} en {$T-}. Als de optie {$T-} gezet is, krijgen we van de adres-operator een adres terug dat we in een veld van het type Pointer kunnen zetten. Als {$T+} gezet is, kunnen we het geleverde adres alleen maar kwijt in een veld van eenzelfde type. Als we toch zouden proberen de waarde aan een niet-passend type over te dragen, dan volgt er een foutmelding van de compiler. Standaard werkt de compiler met {$T-}.

Open parameters

Met deze optie "aan" hoeft het type String dat doorgegeven wordt aan een procedure of functie, niet van hetzelfde type te zijn als in de procedure of functie gedeclareerd is. In de procedure zal de ontvangen string zich gedragen alsof hij wel in de parameterlijst gedeclareerd is. De functie SizeOf geeft bijvoorbeeld de maximale capaciteit van de oorspronkelijke string terug. De optie zet je met {$P+} en {$P-}. Standaard wordt de optie {$P-} aangenomen.

19.5 Fouten opsporen

Debug information

Om de ingebouwde debugger te kunnen gebruiken, moet deze optie "aan" staan. De optie Debug information kan gezet worden met {$D+} of {$D-}. Als de optie "aan" staat, maakt de compiler een lijst waarin de uit te voeren code verbonden wordt met de regelnummers van het programma. Voor deze optie geldt hetzelfde als voor de optie die de bereikcontrole regelt: standaard staat de optie "aan".

Het is verstandig om er een gewoonte van te maken in geteste units de optie {$D-} op te nemen. Als de units voldoende getest zijn, hoeven hier geen fouten meer gezocht te worden. Een dergelijke handelwijze bespaart geheugenruimte. Als de optie {$D+} "aan" staat, en de uitvoering van het programma wordt onderbroken met een foutmelding, dan wordt de cursor geplaatst op de programmaregel die de fout veroorzaakt.

Local symbols

Variabelen en constanten die gedeclareerd zijn in het implementatie-gedeelte van een unit, worden gezien als lokale velden van de unit. Ook constanten en variabelen die gedeclareerd zijn in procedures of functies worden gezien als lokale velden in de procedure of functie. De vorm van de optie is: {$L+} en {$L-}. Als de optie "aan" staat, wordt er bij het compileren van de unit informatie opgeslagen over de lokale symbolen. Met de interne debugger kunnen de waarden uit de diverse lokale velden gelezen worden en kunnen keuzen als «*Call Stack*» worden gemaakt. De in de unit opgeslagen informatie vergroot het aangemaakte .TPU-bestand, maar heeft geen invloed op de grootte en snelheid van een uiteindelijk .EXE-bestand. De optie {$L+} werkt alleen als ook de optie {$D+} "aan" staat. Als de optie {$L-} "aan" staat, kun je met de debugger geen lokale symbolen controleren.

19.6 Inschakelen of nabootsen co-processsor

Numeric processing

De microprocessors 8086, 8088, 80286 en 80386 hebben allemaal een rekenkundig broertje (of zusje?) van het type 8087, 80287 en 80387. Deze rekenkundige co-processors kunnen rekenkundige functies overnemen van de hoofdprocessor. Bij de processors 80486DX en Pentium is zo'n rekenkundige co-processor ingebouwd. Als deze optie, geschreven als {$N+} en {$N-}, "aan" staat, produceert de compiler code die de rekenkundige co-processor aanspreekt. Het spreekt vanzelf dat dit soort programma's alleen kan draaien op machines die met een dergelijke co-processor zijn uitgerust. Met de optie {$N+} beschik je over een viertal extra typen met drijvende puntnotering. In hoofdstuk 4, "Typen en variabelen" is uitgelegd wat dit inhoudt.
Behalve het type Real beschik je met een rekenkundige co-processor ook nog over:

Type:	Bereik:	Bytes:	Nauwkeurigheid in cijfers:
Single	1.5 x 10 -45 tot	4	7-8

	3.4 x 10 38			
Double	5.0 x 10 -324 tot	8	15-16	
	1.7 x 10 308			
Extended	3.4 x 10 -4932 tot	10	19-20	
	1.1 x 10 4932			
Comp	-263+1 tot 263-1	8	19-20	

Het type Comp is bestemd voor de opslag van grote, gehele getallen. Standaard werkt de compiler alsof de optie {$N-} gezet is.

Emulation

De opties {$E+} en {$E-} zorgen voor wel of geen softwarematige nabootsing van een numerieke co-processor als er geen co-processor aanwezig is. Als deze wèl aanwezig is, wordt er code voor de processor geleverd. Standaard werkt de compiler alsof de optie {$E+} gezet is. Als je in je programma {$N+, E+} opneemt, zal het programma een numerieke processor nabootsen, of de aanwezige numerieke processor direct aanspreken. Met de {$E+}-optie kun je ook de extra typen uit de bovenstaande tabel gebruiken, zonder dat je een numerieke processor hoeft te installeren.

19.7 Voorwaardelijk compileren

Conditional defines

Het is mogelijk om de compiler bepaalde voorwaarden te laten testen en deze met bepaalde opdrachten aan de compiler te verbinden. Zo kunnen we bijvoorbeeld stellen: "Als voorwaarde 1 gezet is gebruik dan unit A, gebruik anders unit B". Het programma COMPIL_1 stopt het aanmaken van informatie voor de debugger en de bereikcontrole op arrays als de constante ORGINEEL gedefinieerd is:

{$DEFINE ORIGINEEL}

```
{$IFDEF ORIGINEEL}
{$D-,R-}
{$ELSE}
{$D+,R+}
{$ENDIF}

PROGRAM COMPIL_1;
USES CRT;

BEGIN
   ClrScr;
   GotoXY(20,12);
   {$IFOPT D+}
   Write
   (’Er wordt informatie voor de debugger aangemaakt’);
   {$ELSE}
   Write(’De debug-optie staat af’)
   {$ENDIF};
   Readln
END.
```

Regels: Toelichting:

[1]1 Definieer de constante ORIGINEEL.

```
[2]2-6 Als ORIGINEEL gedefinieerd is, zet dan de optie {$D-}
       en {$R-}, zet anders de opties {$D+} en {$R+}.
[3]7-19Meld op het scherm of de optie {$D+} of {$D-} gezet is.
```

Toelichting:

[1]Om een voorwaardelijke compileeropdracht te kunnen geven, moet er een constante gedeclareerd worden. Deze constante heeft niet een bepaalde waarde, zoals de constanten die we kennen. Dit type constante is wel of niet gedefinieerd.

"{$DEFINE ORIGINEEL}" vertelt de compiler dat de constante ORIGINEEL gedefinieerd is. We kunnen zo'n definitie weer ongedaan maken met "{$UNDEF ORIGINEEL}". Ook dit soort opdrachten aan de compiler wordt tussen accolades geplaatst en voorafgegaan door een dollarteken. $DEFINE werkt alleen op de module waarin het gedefinieerd is.

Als het programma COMPIL_1.PAS units zou gebruiken, dan zou de constante ORIGINEEL in elk van die units apart gedefinieerd moeten worden. De invoerregel "Conditional Defines" onder de keuzen «*Options*» en «*Compiler*», kan gebruikt worden om $DEFINE te zetten. De compileeropdracht «*Build*» verwerkt dan de gestelde voorwaarde in alle modulen die opnieuw gecompileerd kunnen worden. In deze invoerregel blijven accolades en dollartekens achterwege. Je kunt eenvoudigweg de naam van de constante vermelden. Wil je meerdere constanten definiëren, dan worden deze met een puntkomma van elkaar gescheiden.

[2]$IFDEF test of er een constante ORIGINEEL gedefinieerd is. Als dit het geval is, worden de regels uitgevoerd die direct achter $IFDEF staan. Als ORIGINEEL niet gedefinieerd is, worden de regels na $ELSE uitgevoerd. Het geheel wordt afgesloten met $ENDIF. Naast $IFDEF kun je ook nog $IFNDEF gebruiken. Deze opdracht voert een test uit om te zien of een constante niet gedefinieerd is.

[3]In het hoofdprogramma controleert de compiler met $IFOPT of de optie {$D+} gezet is. Als dit het geval is, wordt de schrijfopdracht waarin staat dat er informatie voor de debugger wordt aangemaakt, gecompileerd. Als deze optie niet gezet is, wordt de schrijfopdracht met "De debug-optie staat af" gecompileerd.

Er zijn in Turbo Pascal symbolen voorgedefinieerd. Deze symbolen kunnen bij het compileren getest worden:

```
|Symbolen:          |Betekenis:                                        |
|CPU86              |Geeft aan dat de gebruikte versie van Turbo Pascal |
|                   |bestemd is voor een 80X86 microprocessor.         |
|CPU87              |Geeft aan dat de configuratie die                 |
|                   |gebruikt wordt bij het compileren over een        |
```

	rekenkundige co-processor beschikt.
VER70	Wordt gebruikt om te zien of de Turbo-Pascal versie 7.0 is.
MSDOS	Geeft aan dat de gebruikte versie bestemd is voor het besturingssysteem MS-DOS.

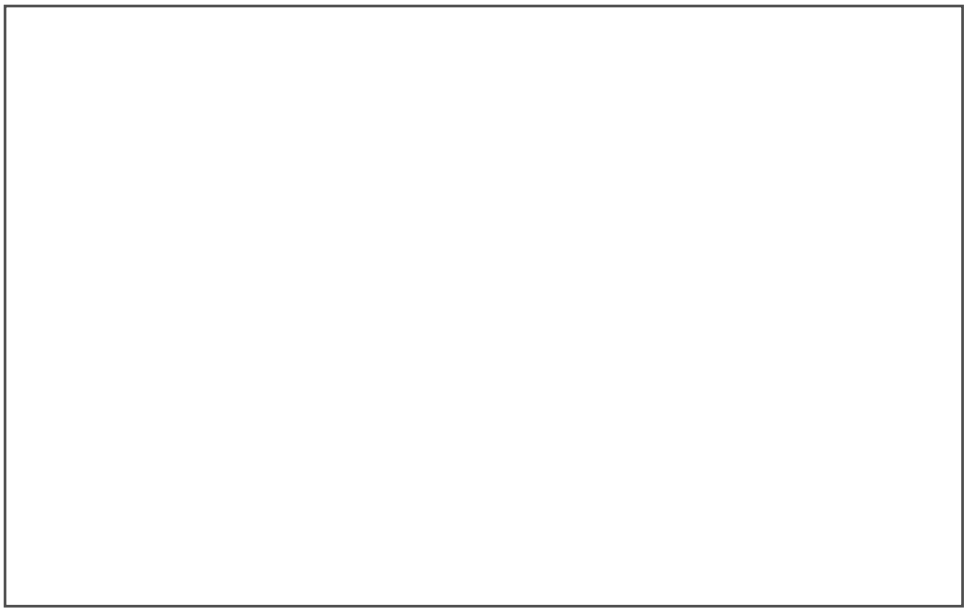
19.8 Instellen van het geheugen

Met de keuzen «*Options*» en «*Memory*» komen we in een invoerscherm waar de verdeling van het geheugen ingesteld kan worden. Dit scherm regelt in feite de $M-optie.

De optie:

{$M 25000, 1000, 655360}

stelt een stack van 25000 bytes in, en geeft aan dat de heap minimaal 1000 bytes groot moet zijn. De omvang van de heap is op het maximum 655360 ingesteld. Als de heap niet minimaal 1000 bytes groot is, zal het programma niet starten. Als de heap op het maximum van 655360 staat, zal de compiler alle overblijvende geheugenruimte toewijzen aan de heap. Een dergelijke regel kan aan het begin van je programma gezet worden. Ook kunnen de afmetingen in het invoerscherm gezet worden.



Afbeelding 19

Als de optie $M niet in het programma opgenomen wordt, en als er ook geen wijzigingen in het invoerscherm zijn aangebracht, geldt standaard de optie:

{$M 16384, 0, 655360}

19.9 Overlays, tekstbestanden en .OBJ-bestanden

Eerder in dit hoofdstuk zijn de opties {$O+} en {$O-} behandeld. Deze opties maken een unit geschikt om gebruikt te worden als overlay. Als de unit daadwerkelijk in een programma als overlay wordt gebruikt, dan wordt dit vlak na de USES-regel aangegeven met de optie {$O <Bestandsnaam>} (zie ook hoofdstuk 10, "Units en overlays").

Je kunt in je programma teksten inlezen die opgeslagen zijn in een ander bestand. De compileeropdracht {$I <Bestandsnaam>} leest op de plaats waar de opdracht staat de tekst in. Deze $I-optie moet je niet verwarren met de opties {$I+} en {$I-}. Deze laatste zijn bestemd om een programma-onderbreking in te stellen ingeval van in- en uitvoerfouten.

Je kunt ook een bestand in assembler aan je Turbo Pascal-programma koppelen. De optie {$L <Bestandsnaam>} leest procedures en functies in die bijvoorbeeld in machinetaal of de programmeertaal C zijn geschreven. Deze optie moet je niet verwarren met {$L+} en {$L-}. Deze laatste zijn bestemd om voor de debugger al dan niet informatie over lokale symbolen aan te maken.

389

389