Pascal uitgediept

Compiler aanwijzigingen

Herman Post
MSX Computer & Club Magazine nummer 71 - november 1994

Deze keer gaat de rubriek wel heel erg diep. In feite wordt nu niet Pascal uitgediept, maar wordt de besturing van de compiler zelf uitgebreid behandeld.
Turbo Pascal kent een aantal opties die invloed hebben op het vertaalproces. Deze opties staan bekend als de compiler directives van de compiler. Hoewel ze in de tekst van een pascalprogramma zijn opgenomen, worden ze niet met een pascalwoord—of statement—aan-of uitgezet. Er is een methode gekozen die er in eerste instantie wat vreemd uitziet. De compiler directives worden opgenomen in een commentaarregel, dus tussen de { en } haken. Zoals gezegd lijkt dit wat vreemd omdat commentaar normaal geen invloed heeft op het programma en deze compiler directives dus wel. Als we er wat verder over nadenken, wordt deze keus echter direct duidelijk. Pascalprogramma's behoren overdraagbaar te zijn naar andere computersystemen en daarom ook overdraagbaar naar andere pascalcompilers. Door de instellingen van de compiler in commentaar te plaatsen, worden deze instellingen door een andere compiler genegeerd en zou het programma overdraagbaar zijn naar een ander systeem. Voor ons is dit niet belangrijk—wie wil er nu een ander computersysteem—zolang we maar weten hoe de compiler directives zijn te gebruiken.

Overzichtslijst directive	default	omschrijving
A	A+	absolute code
B	B+	l/O mode selection
C	C+	CTRL-C and CTRL-S
I	I+	I/O error handling
I	geen	opnemen (includen) van bestanden
R	R-	Index range check
V	V+	parameter (Var) type checking
U	U-	user interrupt
W	W2	nesting of with statements
X	X+	array optimization

Syntax

Een compiler directive is altijd opgebouwd uit een accolade, gevolgd door een dollarteken, een letter en een optie en weer afgesloten met een accolade. Voor of na het dollarteken mag geen spatie staan, maar bij de letter en de optie maakt het niet uit of dit hoofd- of kleine letters zijn. Meerdere compiler directives mogen tussen dezelfde haken achter elkaar staan als ze maar worden gescheiden door een komma. Als u dit doet mag er maar één keer een dollarteken voor staan. Zie hiervoor ook de opmerkingen in het kader.

Enkele voorbeelden zijn:
{$I+}
{$I INCLUDE.FIL}
{$b-,r+,V-}
(*$U+*)

Ik zal alle directives nu stuk voor stuk behandelen. Boven iedere uitleg staat aangegeven hoe de directive normaal— in de default instelling—staat ingesteld als u de compiler opstart. Deze instellingen zijn zo gekozen dat er een goed evenwicht is tussen snelheid en de lengte van de te genereren code.

{$A+}

Deze directive bepaalt of de code die gegenereerd wordt al dan niet absoluut is. Dit is belangrijk bij recursieve procedures of functies. Normaal is een recursieve routine niet mogelijk, maar na een {$A- } is dit wel mogelijk. U kunt dit bij iedere procedure of functie opnieuw instellen en zo alleen routines die recursief zijn met relatieve code genereren.

{$B+}

Met deze directive bepaalt u de selectie van de input/output mode. Deze verwijst bij de normale instelling naar het 'CON' device, en bij de instelling {$B- } wordt verwezen naar het 'TRM' device. Deze compiler directive maakt het mogelijk om Turbo Pascal bij het benaderen van de standaard input/output te laten werken volgens de ISO standaard, welke meer beperkingen oplegt als Turbo Pascal. Deze directive mag alleen voorkomen vooraan in het programma, dus nog voor het declaratiegedeelte, en mag niet tijdens het programma worden gewijzigd. De instelling blijft dus altijd gedurende het hele programma bewaard. Opmerking: via ConlnPtr en ConOutPtr en een eigen afhandeling kan dit wel, zie MCCM 66)

{$C+}

Tijdens een read (ln) enwrite(ln) is het mogelijk om met CTRL| & [s] te pauzeren, en met CTRL & c af te breken. Plaatst u de directive { $C - } dan is dit niet meer mogelijk. Als u de directive uitzet, zal de schermuitvoer iets worden versneld. Deze directive kan net als de B - directive maar één keer in het programma worden geplaatst, en wel voor het declaratie deel. Op onze MSX computer zit er nog een addertje onder het gras. De interrupt verwerkt ook het toetsenbord, en dus ook de

CTRL & [c] aanslag. Het is mogelijk dat het programma hier wordt verlaten, bijvoorbeeld onder DOS2 als de juiste opties aanstaan. Met deze directive moet dus voorzichtig worden omgegaan. Omdat het afbreken van invoer of schermuitvoer meestal toch niet gewenst is, en een beetje extra snelheid bij schermuitvoer leuk is meegenomen, adviseer ik om deze directive altijd uit te zetten.

{$I+}

Hiermee wordt de I/O foutafhandeling gecontroleerd. De controle wordt door Pascal gedaan en resulteert in een I/O error als bijvoorbeeld een file wordt geopend voor lezen, en de file is niet op de disk aanwezig. Als u l $ I - l opgeeft moet u de controle zelf doen aan de hand van de functie IOResult. Hiermee kunt u echter niet de zogenaamde critical error van DOS afvangen. Dus de foutmeldingen van DOS waarop een ABort, Retry or Ignore volgt. Deze zult u moeten afvangen door de DOS errorroutine om te buigen. Ook de Insert disk for drive X: melding is hiermee niet af te vangen, omdat deze vanuit de disk-ROM gebeurt.

{$I FILVAR.INC}

Deze compiler directive heeft geen de-fault waarde. Dit kan ook niet omdat hiermee wordt aangegeven welke sub-file in het programma moet worden opgenomen. De opgegeven naam mag iedere toegestane bestandsnaam zijn. Als er geen extensie wordt opgegeven, dan wordt automatisch '.PAS' toegevoegd. Er hoeft geen spatie te staan tussen de 'I' en de filenaam. Als er geen extensie wordt opgegeven, of een extensie van minder dan drie letters, dan moet er voor de afsluitende accolade een spatie staan.
Nog een aardige opmerking: alle compileropties die in een include-file worden omgezet—behalve de directives met B en C—worden weer teruggezet zodra het programma verder gaat met het compileren van het hoofdprogramma. Hierdoor heeft een include file nooit invloed op de stand van de compiler directives.

{$R-}

De range check die hiermee aan of uit gezet kan worden is bijzonder handig tijdens het ontwikkelen en debuggen van een programma. Met {$R- } wordt niets gecontroleerd, maar met { $R+ } wordt bij iedere benadering van een array gecontroleerd of u wel binnen het bereik van dat array blijft. Het aanzetten van deze directive maakt uw programma veel langzamer, maar voorkomt dat er vreemde fouten ontstaan als u buiten het bereik schrijft. Mijn advies hierbij is daarom om tijdens het ontwikkelen van een programma deze optie altijd aan te zetten, en pas weer uit te zetten bij de definitieve versie.

{$V+}

Als deze compiler directive aan staat wordt bij strings die doorgegeven worden als VAR parameter gecontroleerd of de lengte van de opgegeven string overeenkomt met de lengte die in de functie of procedure heading staat opgegeven. Als u hem uit zet met {$V- } dan wordt hier niet meer op gecontroleerd. Dit is alleen nodig—en handig—als u variabelen over elkaar heen hebt gedeclareerd. Zie MCCM 67. Zet u de controle uit, en weet u niet precies wat u aan het doen bent, dan is de kans groot dat

het programma zichzelf overschrijft, en het hangen van de computer is dan niet uitgesloten. Het advies bij deze optie is dan ook: Alleen gebruiken als u precies weet wat er in het geheugen gebeurt.

{$U-}

Deze directive lijkt erg veel op de C directive wat betreft het gebruik van CTRL & [c]. Het verschil is dat met deze directive ,indien {$U+ } is opgegeven, het programma altijd te onderbreken is, en niet alleen tijdens lees of schrijf operaties. Het aanzetten van de optie maakt uw programma erg veel trager omdat door heel het programma heen op de CTRL & [c] moet worden gecontroleerd. Overigens geldt bij deze optie hetzelfde addertje als bij de optie C. De stand van deze compiler directive is in het programma op te vragen met de boolean-variabele CBREAK. Als de optie actief is {$U+ }, dan kan door het omzetten van de variabele CBREAK het effect teniet worden gedaan. Het programma wordt daar echter niet sneller van, maar de stop-functie wordt wel uitgezet.

{$W2}

Hiermee kan de nesting van het statement WITH worden ingesteld. Het opgegeven getal mag op MSX variëren vanaf l tot en met 9. Het gebruik van een nesting die verder gaat dan twee is overigens alleen nodig bij bijzonder complexe records, dus record binnen record binnen record. Mij is niet bekend of de diepte van nesting invloed heeft op de snelheid van het programma, of op de lengte van de code.

{$X+}

Tot slot de directive die bepaalt hoe array's worden geoptimaliseerd. In de standaard instelling { $X+ } zijn array's geoptimaliseerd voor snelheid. Wordt de directive uitgezet { $X - }, dan genereert de compiler kortere code.

Tot op de bodem

U begrijpt dat de hierboven gegeven informatie erg diepgaand is, en de titel van dit artikel had voor deze keer dan ook beter kunnen zijn 'Pascal uitgediept tot op de bodem'. De informatie in dit artikel heb ik natuurlijk ook niet allemaal in mijn hoofd zitten, maar is afkomstig uit de handleiding en diverse —vooral duitstalige— tijdschriften. Ik heb ook niet de tijd gehad om alle mogelijkheden en beweringen die hierboven staan vermeld uit te proberen en te testen. Het uitzoeken van de gegevens, en het schrijven van dit artikel heeft nu al twee volledige werkdagen gekost, en ook mijn tijd is niet onbeperkt. Heeft u echter opmerkingen of aanvullingen / verbeteringen op het bovenstaande, dan zou ik dat graag van u vernemen.

Extra foutmeldingen

Plaatst u in één directiveregel meerdere aanwijzingen, bijvoorbeeld {$R+,$U-,X+}, dan krijgt u foutmelding 93 en geen verdere uitleg, ook al hebt u de lijst met foutmeldingen bij het opstarten van Turbo Pascal geladen. De foutmelding wordt veroorzaakt door het tweede $-teken; tussen de accolades mag maar één keer, aan het begin, het $-teken staan. De melding komt niet voor in de lijst van foutmeldingen. Deze tekortkoming vindt u in de originele lijst van Turbo Pascal wel vijf keer te-rug, namelijk bij de meldingen:

— 76 Overlays can not be forwarded
— 77 Overlays not allowed in direct mode
— 92 Unable to create overlay file
— 93 Invalid compiler directive
— 96 No nesting of include files

Het is mogelijk om de lijst met foutmeldingen zelf zo aan te passen dat deze meldingen wel worden opgenomen. Ik heb echter tot nu toe geen versie gezien waarbij dit is aangepast. Past u dit aan, dan zal dit ten koste gaan van de vrije ruimte in de editor. Let u vooral op melding 96, omdat deze zelfs niet voorkomt in de handleiding. Het is overigens erg lastig om deze laatste melding te forceren en het is dan ook niet waarschijnlijk dat u deze melding vlug tegenkomt.