'Een **superprogrammeur** doet het werk van drie

Randy Marques gelooft er niets van dat er te weinig goede programmeurs zijn. 'Maar wat gebeurt er met zo iemand: hij of zij maakt carrière en wordt projectleider of architect. Het beroep krijgt te weinig beloning in geld en status.' Even uitgesproken is zijn mening over de kunst van het programmeren. 'Ik vergelijk programmeren met metselen. Je kunt wel een schuurtje in je tuin metselen, maar dan moet je niet denken dat je door te extrapoleren ook een wolkenkrabber kunt bouwen.'

HANS VAN THIEL

PT Embedded Systems: Op je website heb je een link naar een artikel van een van de grondleggers van C, Dennis Ritchie, *The Development of the C Language*. Hij besluit dat artikel, uit 1993, met de zin: 'C is eigenaardig, gebrekkig en een enorm succes'. Dat is wel mooi.

Randy Marques: 'Toen ik er zeventien, achttien jaar geleden voor het eerst mee in aanraking kwam begreep ik C nog niet helemaal. Ik dacht dat het een hogere programmeertaal was. Pas later begon ik me te realiseren dat C eigenlijk een zeer veredelde assembler is. Je moest je druk maken over pointers, conversies, promoties enzovoorts en dat past helemaal niet bij een hogere programmeertaal. Eén van de kenmerken van een hogere programmeertaal is dat je iets maar op één manier kan doen; in C kan je soms hetzelfde op drie manieren doen. Maar het gebruik van C is historisch meegegroeid in een tijd van enorme expansie en, natuurlijk, in samenhang met Unix.'

PT Embedded Systems: Is het voor de goede programmeur niet juist een voordeel dat je iets op drie manieren kan doen?

Randy Marques: 'Nu kom je op een van mijn stokpaardjes! Vaak hoor ik de klacht dat een bedrijf maar geen goede programmeurs kan krijgen. Maar het bedrijf zit eigenlijk vol met goede programmeurs. Maar wat gebeurt er met zo iemand: hij of zij maakt carrière en wordt projectleider of architect. Het beroep krijgt te weinig beloning in geld en status. Ik heb wel eens tijdens een cursus voor projectleiders en architecten gevraagd of ze niet liever zouden programmeren, met hetzelfde salaris en status, en tien van de twaalf zeiden 'ja' op die vraag. Voor velen is programmeur zijn slechts een opstap naar een 'hogere' functie. Dat is jammer, want een superprogrammeur – een programmeur met 8-12 jaar ervaring – kan het werk doen van drie gewone. Eigenlijk twee, maar in de praktijk drie omdat er minder communicatie nodig is.'

PT Embedded Systems: Waarom is er nog steeds standaardisatie nodig voor C? Is het langzamerhand niet af?

Randy Marques: 'De nieuwe standaard biedt additionele functionaliteit in de vorm van onder andere nieuwe types als bool (eindelijk) en complex. Verder hebben we nu variabele lengte-arrays, C++-type comments en kun je op elke plek in je programma variabelen definiëren. Wat ook voortdurende aandacht vereist is de sectie in de standaard die *implementation defined, unspecified* en *undefined behavior* behandelt. In de 1990-standaard staat dat allemaal in appendix F, in de standaard van 1999 is dat Annex J geworden. Annex J is anderhalf keer zo dik als de oude Appendix F. Ik had liever gezien dat dit minder in plaats van meer was geworden, maar ja, standaardisatie is nu eenmaal een democratische aangelegenheid.'

PT Embedded Systems: En op dit moment?

Randy Marques: 'Waar we nu mee bezig zijn, en dat is echt heel essentieel, is de uitbreiding van C voor digitale signaalprocessoren (DSP). Met name mijn collega in de standaardisatiecommissie, Willem Wakker van ACE, heeft zich daar sterk voor ingezet. ACE maakt dan ook dergelijke compilers. De draft voor de uitbreiding is nu klaar en op de volgende bijeenkomst in Curaçao moet hij dan worden bevestigd. In C voor DSP's krijg je onder meer *fixed point arithmetic* en de mogelijkheid om een variabele aan een bepaald fysiek adres te binden [2]. DSP's gebruiken fixed point en hebben allerlei speciale registers. Maar het meeste wat we doen in de standaardisatiecommissie is niet nieuw. Dat is ook niet de bedoeling. De bedoeling is om *existing art* - dat is de term die we gebruiken - zo goed mogelijk eenduidig vast te leggen. Dikwijls is de GNU-compiler in dat verband een goed voorbeeld.'

PT Embedded Systems: Ook in embedded systemen lijkt C terrein te verliezen aan C++, en meer nog aan Java.

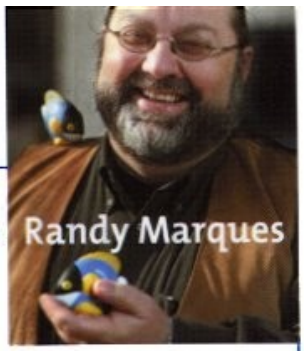
Randy Marques: 'OO talen zijn heel populair, maar ik geloof dat dat meer komt door het pakket aan methoden en de beschikbare programmeeromgeving. Gezien het lage niveau van veel programmeurs – de goeden zijn al projectleider - resulteert dit vaak in het overnemen van een of twee regels uit het boek over gadgets en widgets, en dan in korte tijd iets moois op het scherm zetten. Dit heeft niets met de kennis van de taal te maken. Dit is al- ›

gewone'

'Pas later begon ik me te realiseren dat C eigenlijk een zeer veredelde assembler is'

Foto's: Jan Bogaerts

C en Randy Marques



leen het toepassen van trucjes.
Als je daarentegen je eigen methoden en klassen moet definiëren, dan is dat bepaald geen sinecure.
C++ kan heel krachtig zijn als alles goed is opgezet, maar de taal is behoorlijk complex en vergt zeker een langere leercurve dan C. Het is ook beslist geen superset van C, zoals nog maar al te vaak wordt beweerd. Ooit was dat misschien het geval, maar al ver voor C'99 had je constructies die een verschillende betekenis hebben in C als in C++.
Het is een voortdurende discussie binnen de standaardisatie groepen of C en C++ niet moeten convergeren. Compilerbouwers zijn om praktische reden vaak voorstander daarvan.
Naar mijn mening zijn de uitgangspunten van de talen echter historisch en ook feitelijk verschillend. C++ vertegenwoordigt een overgang naar meer abstractie, en C is bedoeld om dicht bij de machine te blijven en hoogstens de correctheid van programma's te beschermen.'

PT Embedded Systems: Waarom heb je *coding standards* nodig, als je al een ISO-standaard hebt?
Randy Marques: 'Ten eerste vanwege Appendix F dan wel Annex J met implementatieafhankelijk, ongespecificeerd en ongedefinieerd gedrag. Problemen daarmee wil je natuurlijk voorkomen.
Ten tweede heb je wat wij noemen *empirisch vastgesteld wangedrag*. Statistisch onderzoek aan software levert ons de informatie dat bepaalde constructies steeds weer tot fouten leiden. Heel simpel: het vergeten van het tweede 'is gelijk'-teken in een 'if'-constructie waardoor je een assignment krijgt in plaats van een vergelijking.
Nog zo een: als je weet – gebaseerd op de statistiek – dat *multiple returns* dikwijls tot *memory leaks* leiden, dan is het toch beter om geen multiple returns te gebruiken? Het is heel harde informatie die je in een *Coding Standard for C* gebruikt om bepaalde constructies te verbieden, ook al zijn ze legaal volgens de ISO-standaard.
Je ziet dan ook dat de Philips Medical Systems-standaard en de MISRA-coderingsstandaard van de autoindustrie op veel punten volstrekt identiek zijn. Zij zijn niet van elkaar overgeschreven, maar op precies dezelfde informatie gebaseerd.
Daarna heb je dan ook nog de programmeerstijl en daar kan je dan eindeloze en haast religieuze discussies over voeren. Het is met zo'n stijl eigenlijk net als met links of rechts rijden. Het is niet zo belangrijk wat je doet, als je het maar altijd en overal hetzelfde doet.'

PT Embedded Systems: In hoeverre wordt de C'99-standaard door de industrie geaccepteerd? Die is tenslotte nog vrij nieuw, zeker vergeleken met de vorige revisie uit 1990.
Randy Marques: 'Voor *general purpose*-machines als Sun en HP en op universiteiten wordt C'99 langzamerhand wel ingevoerd, gewoon omdat het nieuw is en leuk, maar voor embedded systemen gaat dat een stuk langzamer. Er is ook een goede reden om vast te blijven houden aan C'90. Ontwikkelaars van embedded systemen willen steeds vaker kunnen omschakelen als er een nieuwe chip op de markt verschijnt die twee keer zo snel is en half zoveel kost. Dan is het heel lastig als de compiler voor de nieuwe chip afwijkt van die voor de oude. Vasthouden aan C'90 kan dan heel lonend blijken.
Maar aan de andere kant mis je dan wel de voordelen van bijvoorbeeld variabele lengte-arrays en bijbehorend minder gebruik van de malloc()-functie en als gevolg daarvan weer minder memory leaks! Backwards compatibility is altijd een probleem. Soms moet je bij de standaardisatie gewoon duidelijk vastleggen dat iets nog vijf jaar blijft, maar daarna echt verdwenen zal zijn.'

PT Embedded Systems: Maar als je bestaande code die goed werkt moet gaan herschrijven...
Randy Marques: 'Kijk, ik vergelijk programmeren met metselen. Je kunt wel een schuurtje in je tuin metselen maar dat moet je niet denken dat je alleen maar door te extrapoleren (beetje meer cement, beetje meer bakstenen) ook een wolkenkrabber kunt bouwen. Voor grote gebouwen zet je bijvoorbeeld steigers op die je later weer afbreekt en dat is met software-engineering net zo. Maar projectleiders accepteren zelden dat er code wordt gemaakt om te worden weggegooid.
Aan de andere kant maakt het voor het metselen niet uit of je een fabriek, een ziekenhuis of een flatgebouw metselt. De methodiek is hetzelfde.
Wat anders is dan zeg maar een tien jaar geleden is de toename in volume. Er komen steeds meer systemen met meer dan 1 miljoen regels code. Dit kun je niet meer doen zoals je dat deed voor een project met 10.000 regels code. Helaas wordt er in software alleen de vraag 'Is het mogelijk' gesteld. Zelden de vraag 'Is het verstandig'.
Als je je vak beheerst dan ben je helemaal niet zo bang om een functie opnieuw te schrijven. Waarom zou je bang zijn, je weet toch hoe het werkt? Je bent toch een vakman? Sterker nog, de ervaring leert dat de derde keer zo'n functie op z'n best is ... ◘

Randy Marques is Case-consultant *in product software* (IPS) bij Atos Origin Technical Automation en houdt zich daar bezig met programmeerondersteuning voor embedded software. Hij geeft regelmatig cursussen over C en hij is de enige die geautoriseerd is om de cursus *Safer C* van Les Hatton [1] te geven (in het Nederlands). Eerder werkte Randy Marques bij onder meer Philips Medical Systems aan de *Coding Standard for C*, die binnen Philips en ver daarbuiten als richtlijn wordt gebruikt voor het programmeren van kritische applicaties.
Randy Marques is een van de twee Nederlanders die namens het Nederlandse normalisatie-instituut (NEN) deelneemt aan de internationale standaardisatie van C in de werkgroep ISO JTC1/SC22/WG14. De afkortingen staan achtereenvolgens voor Joint Technical Comittee, Sub Comittee en Working Group. SC22 houdt zich bezig met 'Programming Languages, their Environments and System Software Interfaces'. De werkgroepen behandelen verschillende programmeertalen; WG14 bestrijkt C en WG21 houdt zich bezig met C++. Internationale vergaderingen van deze twee werkgroepen worden 'back to back' gehouden – de ene week C en de andere C++ – zo vertelt Marques, omdat de deelnemersgroepen elkaar sterk overlappen. WG21 is met gemiddeld vijfenzeventig deelnemers wel twee keer zo groot als WG21, wat de huidige populariteit van C++ boven C weerspiegelt.
Er zijn inmiddels twee standaarden voor C in omloop. Die uit 1990 is zeer algemeen geïmplementeerd in compilers maar in 1999 is een nieuwere standaard, C'99, aangenomen door de ISO. Nationale organisaties zoals ANSI (American National Standards Institute) nemen de ISO-standaarden meestal ongewijzigd over, maar geven er wel een eigen naam aan. Zo wordt ook de term 'ANSI C' veel gebruikt.
Het meeste standaardisatiewerk gebeurt via email, maar tweemaal per jaar worden er bij stemming beslissingen genomen op de genoemde bijeenkomsten. De eerstvolgende hiervan wordt door Nederland georganiseerd op Curaçao en Randy Marques is belast met de voorbereiding. Omdat hij zelf van Curaçao afkomstig is, doet het hem veel plezier dat hij met dit evenement iets kan bijdragen aan de internationale beeldvorming over de Nederlandse Antillen.
www.xs4all.nl/~marques/Werk/index.html (bijeenkomst Curaçao), www.jtc1.org (JTC1), http://std.dkuug.dk/jtc1/sc22/ (SC22)

[1] PT Embedded Systems 2000/9, pag. 18-21
[2] PT Embedded Systems 2001/3, pag. 38-41