S T R U C T R A N

V.Batagelj, E.Zakrajšek

FNT - matematika in mehanika

Univerza v Ljubljani, Jugoslavija

Sestavek podaja opis STRUCTRANa - programskega jezika nad FORTRANom s "strukturiranimi" kontrolnimi konstrukti. Zaradi tesne povezanosti s FORTRANom je mogoče STRUCTRAN brez posebnih težav prevesti s predprocesorjem v FORTRAN; za nadaljnje prevajanje pa uporabimo FORTRANski prevajalnik.

STRUCTRAN - In the paper a "structured" language over FORTRAN is described.

UVOD

Že več kot deset let je tega kar se je začelo uveljavljati mnenje, da je nepremišljena raba go to stavka kriva za nepreglednost in nerazumljivost programov. Ta stališča so bila najradikalneje izražena v Dijkstrovem članku [1], v katerem avtor zahteva odpravo go to stavka iz programskih jezikov. Ta zahteva je pri programerjih, ki so poznali v glavnem le COBOL ali FORTRAN, povzročila zmedo in odpor.

Da zahtevano ni nemogoče, sta že prej pokazala Böhm in Jacopini [2] in kasneje Mills [3]; seveda pa je zato treba v jezik vpeljati druge kontrolne konstrukte. Böhm-Jacopinijev izrek trdi, da lahko vsako programsko shemo zapišemo s kontrolnimi konstrukti: while , if then else , ;(sledi) , če operacijam in predikatom sheme dodamo še operacije TRUE , FALSE , POP in predikat TOP (, ki delujejo nad skladom) ter dopustimo dupliciranje kosov programa. Do podobnega rezultata sta prišla tudi Ashcroft in Manna [4], le da sta operacije nad skladom nadomestila z vpeljavo dodatnih spremenljivk, ki hranijo informacijo o poteku izračuna.

Pod vplivom gornjih idej so se pojavili prvi (Algolu podobni) jeziki brez go to stavka. Med njimi je najbolj znan BLISS, ki je prinesel nov jezikovni konstrukt exit, ki je v bistvu nekakšen "go to naprej". BLISS je jezik za pisanje sistemskih programov za računalnike PDP in se je, kakor izvemo v [5,6], dobro obnesel.

Teorijo kontrolne strukture jezikov z exit stavkom so razdelali Peterson, Kasami in Tokura [7]. V omenjenem članku je vpeljana tudi neskončna zanka - jezikovni konstrukt, ki ga srečamo že v [8], in nadomešča "go to nazaj".

Navedimo na kratko najpomembnejše rezultate iz članka [7]:

- s kontrolnimi konstrukti: if then else , ; (sledi) , neskončna zanka in večnivojski exit lahko, če dovoljujemo še dupliciranje, zapišemo vsako programsko shemo. Enonivojski exit za to ne zadostuje.

- če se odpovemo dupliciranju, lahko s preostalimi konstrukti zapišemo natanko vse "pravilno grajene" sheme - programske sheme, ki jih dobimo pri programiranju "od zgoraj navzdol" (top-down).

Poleg preglednosti in lokalnosti ter s tem boljše razumljivosti in lažjega vzdrževanja, dajejo prednost tovrstnim kontrolnim strukturam v programskih jezikih še raziskave na področju dokazovanja pravilnosti [9,10] in optimizacije [11,12] programov.

Ker je pisanje prevajalnikov zamudna stvar, ljudje pa želimo vsako stvar sami preizkusiti, so precej naravna pot do jezikov z opisano kontrolno strukturo predprocesorji, ki iz "strukturiranega" jezika prevajajo v nek višji programski jezik, nad katerim je "strukturirani" jezik zgrajen [13,14,15,16]. Tako smo se tudi sami odločili, da napišemo predprocesor za jezik nad FORTRANom. Poimenovali smo ga STRUCTRAN. Poglejmo si ga pobliže!

SINTAKSA

STRUCTRANovo sintakso bomo podali s takoimenovanimi sintaksnimi diagrami. To so usmerjeni grafi, katerih točke običajno razdelimo na dva razreda: neterminalne in terminalne. Prve grafično predstavimo kot pravokotnike, druge pa kot ovalne like.