Bogdan PRICOP
Gheorghe BĂDĂRĂU

Cap 9. Controlul execuției

- ***Blocuri de instrucțiuni executabile***
- ***Construcția IF***
- ***Instrucțiunile IF logic şi GO TO***
- ***Construcția DO***
- ***Bucle DO simple***
- ***Bucle DO WHILE***
- ***Instrucțiunile GO TO, CONTINUE şi STOP.***

Blocuri de instrucțiuni executabile

O construcție de control constă din unul sau mai multe blocuri de instrucțiuni şi construcții Fortran precum şi o logică a controlului ce cuprinde implicit sau explicit aceste blocuri.

La execuție, în funcție de o condiție de control, se selectează un anumit bloc de instrucțiuni şi construcții.

Un *bloc* este o secvență de zero sau mai multe instrucțiuni de acțiune şi construcții de control. Altfel spus, un bloc este o secvență de construcții executabile. Expresiile din logica controlului execuției determină dacă blocul va fi sau nu va fi executat. De exemplu, la începutul blocului poate exista o instrucțiune de salt ce face ca celelalte instrucțiuni din bloc să nu fie executate; şi în acest caz se consideră că a avut loc o execuție completă a blocului.

Blocuri de instrucțiuni executabile

De obicei construcția de control conține o instrucțiune inițială înaintea unui bloc şi o instrucțiune finală după un bloc. Există construcții ce au mai mult decât un bloc. Construcția include şi condiții ce determină dacă blocul se execută sau nu. Unele din construcții conțin instrucțiuni adiționale ce determină care bloc anume este ales pentru executie.

Pentru blocuri şi controlul blocurilor există următoarele *reguli şi restricții*:

- Când se execută un bloc execuția începe cu prima instrucțiune sau prima construcție a blocului. Instrucțiunile blocului se execută în ordine una după alta atâta timp cât în bloc nu există o construcție de control sau o instrucțiune de control care să modifice ordinea secvențială.

Blocuri de instrucțiuni executabile

- Orice bloc, conceput ca o singură entitate, trebuie să fie complet conținut, înglobat, în interiorul unei construcții.
- În interiorul unui bloc este permis un salt sau o construcție ce trimite controlul la o instrucțiune sau la o construcție din același bloc.
- Ieșirea din bloc se poate face oriunde din interiorul blocului.
- Este interzis saltul din afara blocului la o instrucțiune sau o construcție conținută într-un bloc.
- În interiorul unui bloc sunt permise apelurile unor proceduri.

Construcția IF

O construcție IF selectează pentru execuție cel mult un bloc de instrucțiuni şi construcții. Sintaxa generală a instrucțiunii IF este:

[*nume:*] IF *(expr_1)* THEN
bloc
[ELSE IF *(expr_2)* THEN [*nume*]
bloc] ...
[ELSE [*nume*]
bloc]
END IF [*nume*]

Aici *expr_1, expr_2* sunt expresii logice scalare, *bloc* este un bloc de instrucțiuni şi construcții iar *nume* este nume de construcție IF. Construcția IF conține în mod obligatoriu o instrucțiune IF THEN şi o instrucțiune END IF.

Construcția IF

Opțional ea poate conține instrucțiuni ELSE IF sau o instrucțiune ELSE.

Opțional instrucțiunea IF THEN se poate identifica, sau cum se mai spune, *se poate numi,* cu *nume,* un *nume de* construcţie IF. În acest caz instrucţiunea END IF trebuie să fie şi ea numită să specifice același nume. Dacă instrucțiunea IF THEN nu este numită, nu este numită nici instrucțiunea END IF. Dacă instrucțiunile IF THEN şi END IF sunt numite, este posibil să numim cu același nume instrucțiunea ELSE sau instrucțiunile ELSE IF.

Trebuie luate în considerație următoarele *reguli* şi *restricții:*

- Se execută cel mult unul din blocurile construcției; este posibil ca niciun bloc să nu se execute.

Construcția IF

- După instrucțiunea ELSE nu sunt permise instrucțiuni ELSE IF.
- Sunt interzise salturile la o instrucțiune ELSE IF sau la o instrucțiune ELSE.
- În interiorul unei construcții IF este permis saltul la END IF din oricare din blocurile construcției IF. Saltul din afară la END IF este permis, dar este considerat o caracteristică depășită pe care nu o recomandăm.

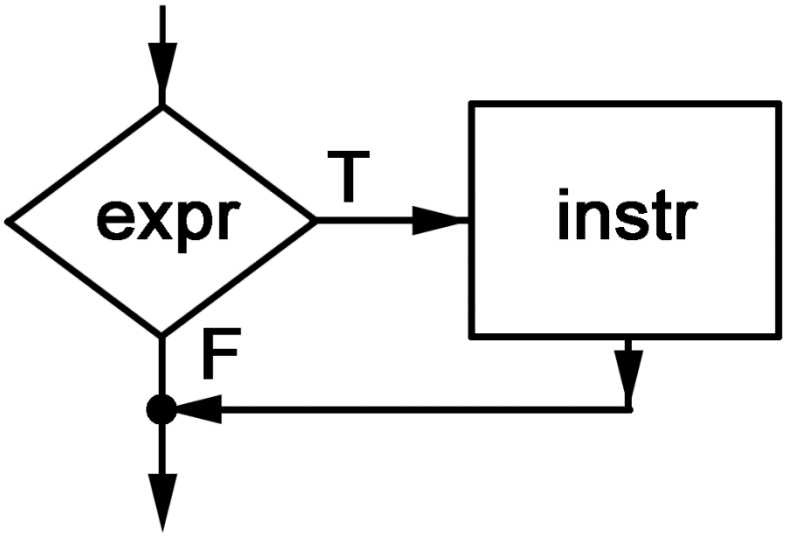
Execuția construcției IF se face astfel: expresiile logice sunt evaluate în ordine până ce găsește una adevărată; atunci se execută primul bloc ce urmează după prima expresie adevărată şi execuția construcției IF se termină. Expresiile logice adevărate care urmează nu mai au niciun efect.

Construcția IF

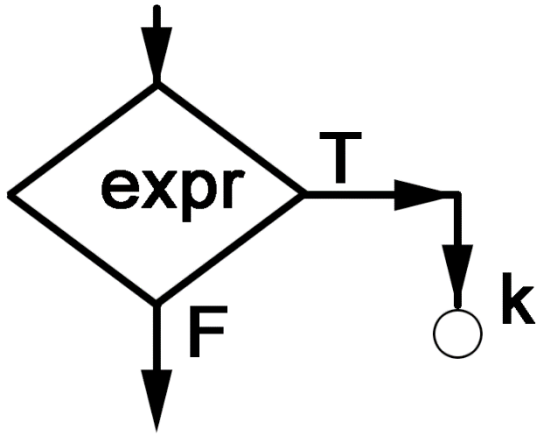
Se poate întâmpla ca niciuna din expresiile logice ale construcției să nu fie .adevărată. în acest caz se execută blocul ce urmează după instrucțiunea ELSE dacă există unul; în caz contrar nu se execută niciunul din blocurile construcției.

Instrucțiunile IF logic și GO TO

Cele mai simple selecții sunt arătate în schemele logice de mai jos. În acest caz nu mai există condițiile ELSE IF sau ELSE şi acțiunea ce trebuie luată dacă expresia logică *expr* este adevărată constă în execuția unei singure instrucțiuni de acțiune *instr* .



a)



b)

Instrucțiunile IF logic și GO TO

În limbajul schemelor logice cerculeţul din figura b este un simbol conector. Simbolul conector se foloseşte atunci când folosirea liniei continue este limitată de dimensiunea hârtiei sau din considerente de estetică a schemei logice. Pentru a identifica conectorii de intrare şi de ieşire se folosesc simboluri de identificare, litere sau cifre. Astfel, conectorii reprezintă un mijloc convenabil pentru a lega două puncte ale schemei logice fără a folosi liniile de flux. O altă funcţie a conectori lor este de a permite reprezentarea pe mai multe pagini a unei scheme logice mai complicate.

Pentru codificarea selecţiei în Fortran există instrucţiunea IF logic. Sintaxa instrucţiunii IF logic este:

IF (expr) *instr*

Instrucțiunile IF logic și GO TO

Aici expresie este o expresie logică scalară iar *instr* este o instrucțiune de acțiune. Instrucțiunea de acțiune nu poate fi o instrucțiune IF sau o instrucțiune END a unui program subrutină sau funcție.

Modul de execuție este următorul: se evaluează expresia logică scalară expr. Dacă valoarea expresiei este .TRUE. se execută instrucțiunea *instr*, după care, controlul trece la instrucțiunea următoare din program. Dacă valoarea expresiei este .FALSE., controlul trece la instrucțiunea ce urmează în program.

Exemplu

instrucțiunea IF logic

```
IF ( val .GT. 0. ) val = 0.0
```

este echivalentă cu următoarea construcție IF:

```
IF (val .GT. 0.) THEN
val = 0.0
ENDIF
```

Instrucțiunile IF logic și GO TO

Cazul din figura b corespunde situației în care instrucțiunea ce urmează a fi executată este instrucțiunea GO TO. Instrucțiunea GO TO este o instrucțiune de salt necondiționat ce afectează ordinea de execuție. Sintaxa instrucțiunii GO TO este următoarea:

GO TO *et*

unde *et* este o etichetă. Eticheta este un șir de 1 până la 5 cifre; zerourile din față se ignoră; aceasta înseamnă că 010, 0010 şi 10 reprezintă aceeași etichetă.

Execuția instrucțiunii GO TO se face astfel: când se execută instrucțiunea GO TO, următoarea instrucțiune ce se va executa este instrucțiunea țintă din aceeași unitate de program instrucțiune ce are eticheta *et*.

Construcția DO

Iterațiile, sau cum se mai numesc, *buclele,* se implementează prin construcții DO. În Fortran există trei forme diferite ale construcțiilor DO:

- construcție DO simplă
- construcție DO WHILE
- construcție DO cu control al iterației.

Există două forme de bază ale construcției DO, DO *cu bloc* și DO *fără bloc* însă practica programării moderne favorizează construcția DO *cu bloc*.

Sintaxa construcției DO este:

[*nume:*] DO [control_buclă]
bloc
END DO [*nume*]

Construcția DO

bloc conține zero sau mai multe instrucțiuni şi construcții a căror execuție repetată este comandată de controlul buclei.

Blocul de instrucțiuni *bloc* se mai numește *domeniul buclei* DO (range). Domeniul buclei DO poate conține şi construcții IF, construcții CASE sau chiar alte construcții DO; este necesar ca orice construcție interioară să fie înglobată complet în construcția exterioară. Dacă domeniul buclei DO conține o altă construcție DO se spune că buclele DO sunt *imbricate* (nested).

Există două instrucțiuni speciale ce pot să apară în domeniul unei construcții DO şi care pot să altereze într-un anumit mod special execuția secvențială a instrucțiunilor şi construcțiilor blocului. Una este instrucțiunea EXIT, cealaltă instrucțiunea CYCLE.

Construcția DO

Instrucțiunea EXIT are sintaxa:

EXIT [*nume*]

Daca în instrucțiunea EXIT este referit un *nume* (un nume de construcție DO), instrucțiunea EXIT trebuie să fie conținută în domeniul acelei bucle DO, altfel instrucțiunea EXIT trebuie să se găsească în domeniul cel puțin a unei bucle DO.

Execuția instrucțiunii EXIT înseamnă terminarea execuției domeniului buclei DO: instrucțiunile de acțiune ce urmează lui EXIT în domeniul buclei DO nu se mai execută şi controlul trece la prima instrucțiune executabilă ce urmează instrucțiunii END DO cu același nume. Dacă EXIT nu este numită, atunci instrucțiunea EXIT termină execuţia buclei DO celei mai interioare.

Construcția DO

Instrucțiunea CYCLE are sintaxa:

CYCLE [*name*]

Dacă în instrucțiunea CYCLE este referit un *nume* (un nume de construcție DO), instrucțiunea CYCLE trebuie să fie conținută în domeniul acelei bucle DO, altfel instrucțiunea CYCLE trebuie să se găsească în domeniul cel puțin a unei bucle DO.

Instrucțiunea CYCLE, spre deosebire de instrucțiunea EXIT, ce termină complet execuția unei construcții DO, întrerupe execuția domeniului şi cauzează începerea unui nou ciclu de execuţie al construcției DO. Cu alte cuvinte, execuţia se întrerupe şi se începe o nouă execuție a domeniului început cu prima instrucțiune a sa. În cazul unei construcții DO cu contor al iterației, după întrerupere se fac şi reajustările necesare la contorul iteraţiei şi la variabila DO.

Construcția DO simplă

În cazul construcției DO simple lipsește controlul buclei. De aceea, buclele DO simple se mai numesc “bucle fără sfârșit„. *Sintaxa* construcţiei DO *simple* este:

[*nume:*] DO
bloc
END DO [*nume*]

Construcția DO simplă, fară controlul buclei, permite să se repete execuția blocului (domeniul buclei) până ce construcția DO se termină în mod explicit la o instrucțiune din interiorul domeniului.

Există două instrucțiuni speciale ce pot să apară în domeniul unei construcții DO şi care pot să altereze execuția secvențială a instrucțiunilor şi construcțiilor blocului. Una este instrucțiunea EXIT iar cealaltă instrucțiunea CYCLE.

Construcția DO simplă

Instrucţiunea EXIT cauzează terminarea imediată a execuţiei construcţiei DO şi transmite controlul la prima instrucţiune executabilă dupa END DO. Sintaxa acestei instrucțiuni este:

EXIT

Instrucțiunea CYCLE, spre deosebire de instrucțiunea EXIT, ce termină complet execuția unei construcții DO, întrerupe execuția domeniului şi cauzează începerea unui nou ciclu de execuție al construcției DO. Sintaxa instrucțiunii CYCLE este:

CYCLE

La întreruperea execuției unei construcții DO de către o instrucțiune CYCLE, variabila de control a iterației este actualizată şi începe procesul de execuție al iterației următoare.

Bucle DO WHILE

Forma *DO WHILE* a construcţiei DO permite execuţia repetată a unui bloc atât timp cât valoarea unei expresii logice rămâne adevărată. Sintaxa instrucţiunii DO WHILE este:

do while (expr)
bloc
end do

Aici *expr* este o expresie logică scalară. Execuţia unei construcţii DO WHILE se face astfel: se evaluează expresia logică *expr*; dacă valoarea expresiei este .TRUE. se execută blocul.

După execuţia ultimei instrucţiuni a blocului, execuţia revine la instrucţiunea DO WHILE, expresia logică *expr* este din nou evaluată şi ciclul se repetă. Dacă valoarea expresiei logice *expr* este .FALSE. blocul nu se mai execută şi controlul trece la prima instrucţiune executabilă după END DO.

Construcția DO cu contor al iterației

În acest caz o variabilă întreagă ic, numită *contorul iterației*, controlează de câte ori se execută blocul buclei.

Sintaxa construcţiei DO cu contor al iterației este următoarea:

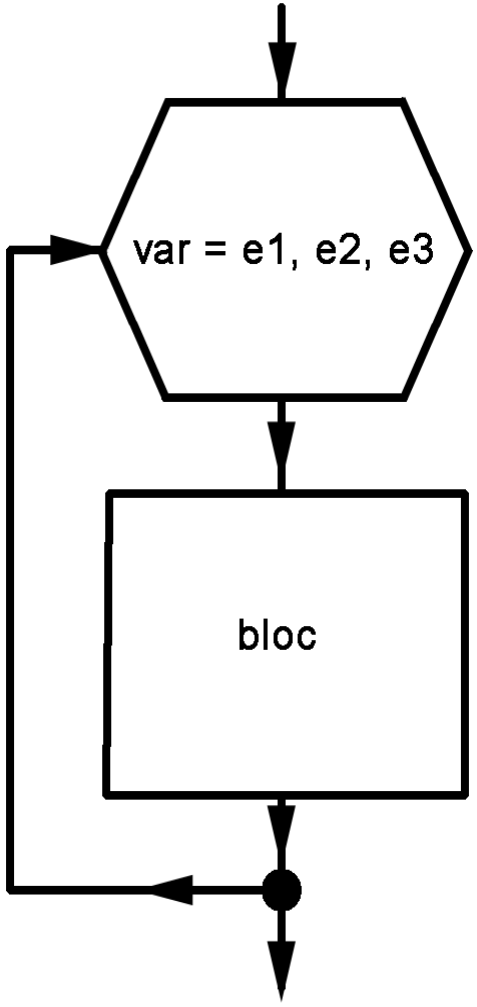
[*nume:*] DO i = e1, e2 [, e3]
bloc
END DO [*nume*]

Aici i = e1, e2 [,e3] se numeşte *controlul iteraţiei*. Mărimea i, numită *variabilă* DO, sau *variabila buclei*, este o variabilă întreagă sau reală, iar e1, e2 şi e3 sunt expresii numerice întregi sau reale. Dacă e3 este 1 se poate omite scrierea lui.

Construcția DO cu contor al iterației

Acest tip de construcție este des întâlnit în programare, motiv pentru care a fost introdus un simbol special, simbolul preparare, a cărui schemă este prezentată în dreapta.

După terminarea execuției construcției DO, variabila DO își păstrează ultima valoare, cea pe care a avut-o atunci când a fost testat contorul iterației ic. În decursul execuției domeniului buclei, variabila DO nu trebuie redefinită, nici nu trebuie să rămână nedefinită. Observăm că schimbarea în decursul execuției blocului a variabilelor folosite în expresiile e1, e2 sau e3 nu modifică contorul iteraţiei; acesta este fixat la intrarea în iteraţie.



Bibliografie

- *Octavian PETRUȘ, Fortran 90/95, Limbaj și Tehnici de programare, Editura Universității Tehnice "Gheorghe Asachi" din Iași, 2001*
- Romeo CHELARIU, Sisteme de operare şi limbaje de programare (Îndrumar de laborator), http://www.sim.tuiasi.ro/wp-content/uploads/Chelariu-indrumar-solp.pdf, 2004
- https://ro.wikipedia.org