Mr. Mirko Varga
Fakultet organizacije i informatike
Varaždin

UDK: 681.3.06
Pregledni rad

Razumijevanje i korištenje COBOL standarda ANSI 85

U radu se obradjuje problematika razvoja programskog jezika COBOL s posebnim osvrtom na novosti koje uvodi najnoviji standard ANSI 85. Ovaj standard uvodi niz novina koje se odnose na domenu strukturnog programiranja, proširenja osnovnog skupa naredbi uz uvodjenje nekih novih naredbi koje su standardne u nekim drugim programskim jezicima, kao npr: struktura "CASE" ("EVALUATE" - "END EVALUATE" u COBOL-u). Osim toga raspravlja se takodjer o poboljšanoj interprogramskoj komunikaciji, olakšanom pristupu podstringovima te novoj metodi specifikacije varijabilne dužine sloga.

Ključne riječi: COBOL, ANSI 85, standard programiranja.

1 Uvod

Posljednja verzija COBOL standarda raspoloživa je od 1985. godine, a poznata je pod nazivom ANSI 85. Ona je usmjerena podizanju učinkovitosti niza faktora, prvenstveno povećanju produktivnosti programiranja, smanjenju troškova u eksploataciji, odnosno održavanju te većoj portabilnosti programa.

COBOL ANSI 85 uvodi mnoga poboljšanja, a najznačajnija su slijedeća:

- proširenje skupa naredbi strukturnog programiranja
- interprogramska komunikacija
- uvodjenje naredbi "INITIALIZE" i "REPLACE"
- omogućen pristup podstringovima
- fleksibilni oblik varijabilne dužine sloga

Revizija COBOL-a vodila je računa takodjer o kompatibilnosti s već egzistirajućim kodom ranije napisanih COBOL programa. Maksimalizacija dobiti i minimalizacija troškova jest orijentacija ANSI 85 standarda. Potencijalni troškovi koncentrirani su na konverziju koda starih programa te troškovima obrazovanja u cilju prihvaćanja i primjene novog standarda.

2 Karakteristične naredbe strukturnog programiranja

2.1 Naredba IF

```
1 IF C1
2 THEN IF C2
3      THEN S1
4      ELSE S2
5      END-IF
6 ELSE S4
7 END-IF
```

Sl. 1 Primjer strukturne naredbe IF

Primjer strukturne naredbe IF prikazuje slika 1. END-IF na liniji 5 terminira naredbu IF koja počinje na liniji 2, ali ne terminira naredbu IF koja počinje na liniji 1. Opcija THEN doprinosi boljoj čitljivosti programa. Problem terminiranja "ugnježdenih" IF naredbi mogu pričinjavati naredbe s uvjetnom opcijom, primjerice READ s opcijom AT END; ADD, SUBTRACT, MULTIPLY, DIVIDE s opcijom ON SIZE ERROR itd. Ove naredbe imaju zato posebne terminatore završetka. Kompletnu listu terminatora prikazuje tabela 1.

END-ADD	END-IF	END-SEARCH
END-CALL	END-MULTIPLY	END-START
END-COMPUTE	END-READ	END-STRING
END-DELETE	END-RECEIVE	END-SUBTRACT
END-DIVIDE	END-RETURN	END-UNSTRING
END-EVALUATE	END-REWRITE	END-WRITE

Tab. 1 C85 Terminatori

2.2 Naredba "CONTINUE"

```
1 IF C1
2       IF C2
3           CONTINUE
4       ELSE
5           S2
6       END-IF
7       S3
8 ELSE
9       S4
10 END-IF
```

Sl. 2 Primjer korištenja naredbe CONTINUE

Primjer korištenja naredbe CONTINUE u strukturnoj naredbi IF prikazuje slika 2. Ukoliko je zadovoljen uvjet C2, prelazi se na tzv. nul-naredbu CONTINUE koja se tretira kao neizvršna naredba (primjerice kao u programskom jeziku FORTRAN gdje se koristi za označavanje kraja DO petlje) tj. kontrola se prenosi na prvu naredbu iza naredbe END-IF, odnosno izvršava se naredba na liniji 7. Ova naredba doprinosi boljoj preglednosti programa i zamjenjuje zapravo standardnu naredbu NEXT SENTENCE jer njezino korištenje nije preporučljivo u kombinaciji sa strukturnom naredbom IF, END-IF.

2.3 Naredba "EVALUATE"

```
1 EVALUATE TRUE
2       WHEN C1 THEN S1
3       WHEN C2 THEN S2
4       WHEN C3 THEN S3
5       WHEN OTHER THEN S9
6 END-EVALUATE
```

Sl. 3 Primjer CASE strukture

U programskom jeziku COBOL već dugo se osjećala potreba za uvodjenjem naredbe koja omogućuje višestruki izbor a koja je zastupljena u suvremenim programskim jezicima, primjerice u PASCAL-u pod nazivom "CASE". Analogna naredba u COBOL-u je naredba "EVALUATE" sa sličnom strukturom. Dakle, navode se uvjeti i odredjene akcije koje treba poduzeti ako je pojedini uvjet zadovoljen. Konkretno na slici 3 izvodi se akcija S1 ako je zadovoljen uvjet C1, akcija S2 ako je ispunjen uvjet C2, akcija S3 ako je istinit uvjet C3, a u slučaju da ni jedan od spomenutih uvjeta (C1, C2, C3) nije ispunjen, izvodi se naredba, odnosno sekvenca naredbi označena sa S9. Nakon bilo koje akcije kontrola se prenosi na prvu naredbu iza naredbe "END-EVALUATE".

2.4 Naredba "PERFORM"

```
1 PERFORM UNTIL C1
2       S1
3       ...
10      S9
11 END-PERFORM
```

Sl. 4 Struktura "pitaj pa radi"

```
1 PERFORM WITH TEST AFTER UNTIL C1
2       S1
3       ...
10      S9
11 END-PERFORM
```

Sl. 5 Struktura "radi pa pitaj"

Na slikama 4 i 5 dat je prikaz tipičnih struktura koje se realiziraju preko vrlo moćne naredbe PERFORM. Osnovna razlika izmedju njih sastoji se u tome da li se testiranje uvjeta vrši prije ili nakon izvodjenja sekvence naredbi S1 do S9, tj. spomenuta sekvenca na slici 5 izvršit će se najmanje jednom za razliku od koda na slici 4 gdje se može dogoditi da se naznačena sekvenca uopće ne izvrši jer nije ispunjen uvjet C1. Naravno, izvršne naredbe S1 do S9 mogu uključivati uvjetne naredbe kao i sve oblike naredbe PERFORM.

3 Interprogramska komunikacija

```
1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. PROG1.
3     .....
4     .....
5 CALL "PROG2".
6     .....
7     .....
8 CALL "PROG3".
9     .....
10    .....
11 IDENTIFICATION DIVISION.
12 PROGRAM-ID. PROG2.
13    .....
15    .....
20 END PROGRAM PROG2.

21 IDENTIFICATION DIVISION.
22 PROGRAM-ID. PROG3.
23    .....
24    .....

50 END PROGRAM PROG3.
```

Sl. 6 Primjer "ugnježdenih" programa

Novi standard poboljšao je i mogućnosti u pogledu interprogramske komunikacije COBOL programa. Tipični razlozi potrebe za interprogramskom komunikacijom programa dali bi se svesti na slijedeće uvjete:

- "ugnježđeni" programi, odnosno moduli koji su definirani unutar samog programa ili nekih drugih programa

- definiranje globalnih entiteta (podataka i datoteka) u jednom programu uz mogućnost višestrukog korištenja od strane različitih programa

- omogućavanje pristupa eksternim entitetima (podacima i datotekama) od strane bilo kojeg programa u izvodjenju

- postojanje posebne skupine programa tzv. inicijalnih programa čiji je isključivi zadatak postavljanje odnosno (re)inicijalizacija odredjenih varijabli neposredno prije poziva programa

Primjer "ugnježđenih" programa prikazan je na slici 6. Program P1 objedinjava dva programa P2 i P3. Svaki od ovih programa završava sa specifičnom naredbom END PROGRAM. Program P1 poziva programe P2 i P3 pomoću standardne naredbe CALL koja se uobičajeno koristi u više programskih jezika za poziv potprograma, odnosno specijalnih sistemskih rutina. Program P2 može pozivati P3 dok P3 ne može pozivati niti jedan program.

Globalni entiteti koji se većinom koriste u poslovnim obradama jesu odgovarajuće strukture podataka: slogovi, datoteke, baze podataka, standardni izvještaji, odnosno tipični upiti koji se odnose na odgovarajuće baze podataka. Njihovo definiranje provodi se specificiranjem klauzule GLOBAL. Konkretno, ukoliko je u našem primjeru datoteka definirana kao globalni entitet u programu P1, njoj mogu pristupati programi P1, P2 i P3.

Eksternim entitetima pridaje se klauzula EXTERNAL kojom je omogućen pristup odredjenom eksternom entitetu od strane bilo kojeg programa u izvodjenju. Koncept globalnih i eksternih entiteta omogućuje dakle konvencionalni pristup, odnosno pristup entitetima bez obzira koji se program trenutno nalazi u izvodjenju.

4 Naredba "INITIALIZE"

```
1 01 TABELA-KUPCA.
2       03 PROIZVOD OCCURS 999.
3          05 SIFRA PIC X(7).
4          05 NARUDZBA PIC S9(9).
5          05 ISPORUKA PIC S9(9).
           .....
           .....
7       INITIALIZE TABELA-KUPCA
8          REPLACING NUMERIC DATA BY ZERO.
```

Sl. 7 Primjer naredbe "INITIALIZE"

Posebna skupina programa za inicijalizaciju koristi klauzulu INITIAL koja se definira u PROGRAM-ID paragrafu. Primjer naredbe za inicijalizaciju prikazuje slika 7. Kada je ona specificirana, podaci koje ona zahvaća poprimaju pretpostavljene ("default") vrijednosti, tj. numerički tipovi podataka vrijednost nula, a nenumerički skup praznina ("spaces") zavisno od veličine deklariranog polja ukoliko nije drugačije deklarirano naredbom REPLACING.

Standardna kategorizacija podataka u COBOL-u razlikuje slijedećih pet kategorija.

1. alfabetske ("ALPHABETIC")
2. alfanumeričke ("ALPHANUMERIC")
3. numeričke ("NUMERIC")
4. alfanumeričke editirane ("ALPHANUMERIC-EDITED")
5. numeričke editirane ("NUMERIC-EDITED")

Vrijednost koja će biti korištena za inicijalizaciju navodi se nakon ključne riječi REPLACING, tj. obično je to rezervirana riječ (u našem primjeru "ZERO") ili literal.

5 Varijabilna dužina sloga

```
1 FD DATOTEKA-KUPACA
        RECORD IS VARYING IN SIZE FROM 10 TO 80
        DEPENDING ON DD-RECORD-LENGTH.
2 01 SLOG-KUPCA.
3          03 SIFRA PIC X(6).
4          03 FILLER PIC X(74).
5 01 DD-RECORD-LENGTH PIC 999.
6 01 LENGTH-OF-CURRENT-RECORD PIC 999.

7 MOVE LENGTH-OF-CURRENT-RECORD
              TO DD-RECORD-LENGTH.
8 WRITE SLOG-KUPCA.
```

Sl. 8 Primjer specifikacije varijabilne dužine sloga

Primjena nove metode specificiranja varijabilne dužine sloga ilustrirana je na slici 8. Dužina sloga kupca može varirati od 10 do 80 znakova za što se koristi fraza DEPENDING ON kojom se to označava. Polje DD-RECORD-LENGTH sadrži stvarnu dužinu tekućeg sloga te se navodi izvan FILE SECTION dijela programa.

Dužina svakog pojedinog sloga posebno se izračunava tj. prije samog zapisa sloga (instrukcija 8) prenosi se stvarna dužina sloga u polje DD-RECORD-LENGTH (instrukcija 7) . Primjena slogova varijabilne dužine posebno dolazi do izražaja kod datoteka s indeksnom i relativnom organizacijom. Naravno da se ovakva mogućnost obilato koristi kod sortiranja i uparivanja datoteka ("SORT", "MERGE") .

6 Zaključak

Tradicionalna struktura COBOL programa iziskivala je od programera dosljedno poštivanje hijerarhijske organizacije samog jezika. Naravno da se to direktno odražavalo na veličinu i redundanciju programa čiji je kod u usporedbi s kodom u drugim programskim jezicima obično bio daleko veći.

Novi standard daleko fleksibilnije pristupa ovoj problematici te uvodi niz novina koje će razveseliti programere odane ovom programskom jeziku. U prvom redu naglasimo činjenicu da neke strukture jezika ("division", "section", "paragraph", "label" i sl.), koje su ranije bile obavezne, postaju fakultativne.

Elementi jezika koji poprimaju fakultativnu odnosno opcionalnu ulogu su slijedeći:

- ENVIRONMENT DIVISION
- CONFIGURATION SECTION
- SOURCE-COMPUTER paragraph
- OBJECT-COMPUTER paragraph
- DATA DIVISION
- LABEL RECORD clause
- FILLER
- PROCEDURE DIVISION

Iz navedenog se može zaključiti da se COBOL program u novom "svjetlu" koji ima samo WORKING-STORAGE SECTION bez PROCEDURE DIVISION može bez problema prevoditi i izvršavati, što je ranije bilo nezamislivo. Osim toga, novi standard uvodi 57 novih rezerviranih riječi.

Literatura

[1] S. Alagić, Principi programiranja, Svjetlost, Sarajevo, 1976.

[2] I. Kononenko, Programski jeziki, Didakta, Radovljica, Ljubljana, 1992.

[3] R. Steven, ANSI 85: Understanding and using the Latest COBOL Standard, Computer Language 10 (3), 67-78 (1993).

[4] R. Sethi, Programming languages, Concepts and Constructs, Addison-Wesley, 1989.

[5] S. Tkalac, Uvod u Cobol, Informator, Zagreb, 1983.

[6] M. Žugaj, M. Varga, Neke mogućnosti podizanja produktivnosti programiranja u elektroničkoj obradi podataka, Zbornik radova IV naučnog skupa pod nazivom "Proučavanje i mjerenje rada" - PROMJER '87, Mašinski fakultet Mostar, 28-30.05.1987., str. 303-314.

Primljeno: 1993-07-09

Varga M. Understanding and using the latest COBOL standard ANSI 85

SUMMARY

ANSI COBOL standard 85 is not fully compatible with ANSI 74. The benefits to be gained by taking advantage of the new language features are many. Cleaner, better structured through the use of the structured programming facilities, more readable, and more maintainable code is one of the more significant benefits. The changes made for the C85 standard show that this version of COBOL will be around for quite some time. The standards committee is currently meeting to solicit input for an object-oriented version of the programming language.