

ZAM-41

Język programowania

cobol

INSTYTUT MASZYN MATEMATYCZNYCH „BOINTE”

Język programowania

COBOL

dla ZAM 41

Opis

WYDAWNICTWA PRZEMYSŁU MASZYNOWEGO „W E M A”
WARSZAWA 1972

Opracował Zespół:

Krystyna Niemiec-Pomaska
Zofia Okraśńska
Janina Pawlikowska-Kozłowska

Komitet Redakcyjny: J. Borowiec (red. nacz.),
W. Kossakowski, A. Mazurkiewicz,
J. Wierzbowski, A. Wiśniewski,
H. Drozdowska (sekr. red.)

Adres redakcji: Warszawa, ul. Krzywickiego 34,
tel. 28-37-29

WPM "WEMA". Warszawa 1972. Wyd. I. Nakład 2.500+45 egz.
Ark. wyd. 13, 18. Ark. druk. 15, 17. Papier offset III kl. 80g,
A1. Format 100x197 mm. Zam. 667/72-5-Z/S.

SPIS TREŚCI

str.

1. WSTĘP	13
1.1. KRÓTKA CHARAKTERYSTYKA JĘZYKA COBOL	13
1.2. IMPLEMENTACJA DLA MASZYNY ZAM 41 . .	16
1.3. UKŁAD PODRĘCZNIKA	16
2. OGÓLNE WIADOMOŚCI O JĘZYKU COBOL	19
2.1. PRZEZNACZENIE I TIPOWE ZASTOSOWANIA	19
2.2. PROGRAM ŹRÓDŁOWY A PROGRAM WYNIKOWY, TRANSLATOR, IMPLEMENTACJA JĘZYKA . .	22
2.3. OGÓLNA STRUKTURA PROGRAMU COBOL . .	23
2.4. KONCEPCJA ORGANIZACJI DANYCH W CO- BOL-u	26
2.4.1. Numer poziomu	30
2.4.2. Specjalne numery poziomu	34
2.4.3. Kwalifikowanie nazw	35
3. POJĘCIA PODSTAWOWE .	37
3.1. METAJĘZYK	37
3.2. DEFINICJE POJĘĆ PODSTAWOWYCH	41
3.2.1. Pojęcia podstawowe dotyczące pro- gramu	41
3.2.1.1. Zbiór znaków języka	41
3.2.1.2. Słowo	42
3.2.1.2.1. Identyfikator	42
3.2.1.2.1.1. Nazwa jednostki danych . . .	43
3.2.1.2.1.2. Nazwa warunku	43
3.2.1.2.1.3. Nazwa wskaźnika	44

	str.
3.2.1.2.1.4. Nazwa procedury	44
3.2.1.2.1.5. Słowo zastrzeżone	44
3.2.1.2.1.6. Stała figuratywna	45
3.2.1.2.1.7. Nazwy specjalnych rejestrów TALLY i DRUM	46
3.2.1.2.1.8. Nazwa pozycji metrykowej.	47
3.2.1.2.1.9. Nazwa zbioru	47
3.2.1.2.1.10. Nazwa biblioteczna	48
3.2.1.2.1.11. Zmienne zastrzeżone BUFF i HARDWARE	48
3.2.1.2.2. Stała	49
3.2.1.2.2.1. Liczba	50
3.2.1.2.2.2. Tekst	52
3.2.1.2.2.3. Numer poziomu	55
3.2.1.2.2.4. Obraz	55
3.2.1.2.3. Punkt ponawiania programu	55
3.2.2. Pojęcia podstawowe dotyczące da- nych	56
3.2.2.1. Zbiór znaków dla danych	56
3.2.2.2. Zbiór i dokument	56
3.2.2.2.1. Własności fizyczne zbioru	57
3.2.2.2.2. Własności logiczne zbioru	58
3.2.2.2.3. Hierarchia danych w zbiorze	58
3.2.3. Program źródłowy i program wyni- kowy	58
3.2.3.1. Pole dokumentu	59
3.2.3.2. Bufor	59
3.3. ORGANIZACJA DANYCH	60
3.3.1. Hierarchia danych	60
3.3.2. Organizacja pamięci operacyjnej dla danych oraz miejsc robo- czych	63

3.3.2.1. Struktura pamięci	63
3.3.2.2. Pole pozycji; postać danych w pamięci	64
3.3.2.3. Pole dokumentu	67
3.3.2.4. Bufor zbioru	68
3.3.3. Dane na nośnikach	70
3.3.3.1. Dane związane z nośnikiem pa- pierowym	70
3.3.3.1.1. Zbiór znaków dla danych . . .	70
3.3.3.1.2. Struktura danych	70
3.3.3.1.3. Ograniczniki danych na nośni- kach papierowych	73
3.3.3.1.4. Dokumenty specjalne	75
3.3.3.1.4.1. Metryka początku zbioru . .	75
3.3.3.1.4.2. Oznacznik operacji ponawia- nia programu	76
3.3.3.1.4.3. Zerowator licznika dokumen- tów	77
3.3.3.1.5. Zbiory pozycyjne	77
3.3.3.1.6. Zbiory niepozycyjne	79
3.3.3.1.6.1. Rozpoznawanie typu dokumen- tu zbioru niepozycyjnego . .	82
3.3.3.2. Postać danych na taśmach magne- tycznych	86
3.3.3.2.1. Postać i organizacja zbiorów taśmowych	86
3.3.3.2.2. Metryka początku zbioru . . .	90
3.3.3.2.3. Metryka końca zbioru	92
3.3.3.2.4. Metryka początku szpuli . . .	92
3.3.3.2.5. Metryka końca szpuli	93
3.3.3.2.6. Metryka końca zapisu	93
3.3.3.2.7. Blok RERUN	93

str.

4. BUDOWA PROGRAMU	95
4.1. IDENTIFICATION DIVISION - IDENTYFI-	
KACJA PROGRAMU	95
4.2. ENVIRONMENT DIVISION - URZĄDZENIA . .	96
4.2.1. CONFIGURATION SECTION - sekcja	
konfiguracji	96
4.2.1.1. SWITCH - opis wskaźnika	97
4.2.1.2. Zwrot CURRENCY	98
4.2.2. INPUT..OUTPUT SECTION - SEKCJA	
WEJŚCIA-WYJŚCIA	99
4.2.2.1. Paragraf FILE..CONTROL deklaracja	
SELECT	100
4.2.2.2. Paragraf I..O..CONTROL	107
4.2.2.2.1. Zwrot APPLY	107
4.2.2.2.2. Zwrot SAME	113
4.2.2.2.3. Zwrot MULTIPLE	114
4.2.2.2.4. Zwrot RERUN	115
Wkładka I	120
4.3. DATA DIVISION - DANE	126
4.3.1. FILE SECTION - SEKCJA ZBIORÓW . .	126
4.3.1.1. FD - opis zbioru danych	126
4.3.1.2. SD - opis zbioru sortowania . .	127
4.3.1.3. Zwrot DATA	128
4.3.1.4. Zwrot LABEL	129
4.3.1.5. Zwrot VALUE OF	130
4.3.1.6. Zwrot BLOCK	132
4.3.1.7. Zwrot RECORD	133
4.3.1.8. Opis dokumentu	133

4.3.1.8.1. Zwrot FIXED	135
4.3.1.8.2. Zwrot FIXED2	135
4.3.1.8.3. Zwrot FLOAT	135
4.3.1.8.4. Zwrot PICTURE	136
4.3.1.8.5. Zwrot BLANK	147
4.3.1.8.6. Zwrot JUSTIFIED	147
4.3.1.8.7. Zwrot VALUE	148
4.3.1.8.8. Zwrot OCCURS	149
4.3.1.8.9. Zwrot REDEFINES	150
4.3.1.8.10. Deklaracja nazwy warunku	151
4.3.2. WORKING STORAGE SECTION - SEKCJA ROBOCZYCH	152
Wkładka II	157
4.4. PROCEDURE DIVISION - PROCEDURY.	164
4.4.1. Struktura części proceduralnej.	164
4.4.2. Instrukcje i zdania, paragrafy i sekcje	164
4.4.3. Reguły przestankowania	166
4.4.4. Segmentacja programu	166
4.4.5. Zdanie warunkowe, bezwarunkowe i sterujące translacją	168
4.4.6. Wyrażenia arytmetyczne i warunki. Kwalifikowanie i indeksowanie nazw	170
4.4.6.1. Wyrażenia arytmetyczne	170
4.4.6.2. Warunki	172
4.4.6.2.1. Porównanie wielkości	172
4.4.6.2.2. Sprawdzenie klasy	174
4.4.6.2.3. Nazwa warunku	174
4.4.6.2.4. Nazwa wskaźnika	175

str.

4.4.6.2.5. Działania logiczne	175
4.4.6.2.6. Moc i kolejność wykonywania działań występujących w warunkach	176
4.4.6.3. Kwalifikowanie nazw	176
4.4.6.4. Indeksowanie nazw	178
4.4.7. OPIS ZDAŃ ENTER, NOTE, EXIT, USE, COPY, END	181
4.4.7.1. Zdanie ENTER	181
4.4.7.2. Zdanie NOTE	182
4.4.7.3. Zdanie EXIT	183
4.4.7.4. Zdanie USE	184
4.4.7.5. Zdanie COPY	189
4.4.7.6. Zdanie END	191
4.4.8. Instrukcje języka COBOL	191
4.4.8.1. Instrukcje związane z wejściem- wyjściem	191
4.4.8.1.1. Instrukcja OPEN	191
4.4.8.1.2. Instrukcja READ	195
4.4.8.1.3. Instrukcja WRITE	198
4.4.8.1.4. Instrukcja CLOSE	199
4.4.8.1.5. Instrukcja ACCEPT	204
4.4.8.1.6. Instrukcja DISPLAY	206
4.4.8.2. Instrukcje obliczeniowe i prze- słania	207
4.4.8.2.1. Instrukcja COMPUTE	207
4.4.8.2.2. Instrukcja MOVE	208
4.4.8.2.3. Instrukcja EXAMINE	212
4.4.8.3. Instrukcje sterujące	214

	str.
4.4.8.3.1. Instrukcja GO TO	214
4.4.8.3.2. Instrukcja IF	215
4.4.8.3.3. Instrukcja ALTER	217
4.4.8.3.4. Instrukcja PERFORM	218
4.4.8.3.5. Instrukcja STOP	220
4.4.8.4. Instrukcje związane z sortowa- niem	221
4.4.8.4.1. Instrukcja SORT	221
4.4.8.4.2. Instrukcja RELEASE	227
4.4.8.4.3. Instrukcja RETURN	227
4.4.8.5. Instrukcje pomocnicze	228
4.4.8.5.1. Instrukcja READY TRACE	229
4.4.8.5.2. Instrukcja RESET TRACE	230
4.4.9. Język wstawek	230
4.4.9.1. Podstawowe jednostki syntaktycz- ne	230
4.4.9.2. Opis języka	236
4.4.9.2.1. Rozkaz	236
4.4.9.2.2. Dyrektywa REZE	237
4.4.9.2.3. Komentarz	237
4.4.9.2.4. Wiersz wstawki	238
4.4.9.2.5. Wstawka	238
Wkładka III	241
5. WIADOMOŚCI UZUPEŁNIA- JĄCE	251
5.1. ZBIORY BIBLIOTECZNE	251
5.1.1. Korzystanie z Biblioteki COBOL	251
5.1.2. Tworzenie zbiorów bibliotecz- nych	252

5.2. RERUN - PONAWIANIE PROGRAMU	255
5.3. JĘZYK OPERACYJNY	259
5.3.1. Dyrektywy języka operacyjnego COBOL-u	259
5.3.1.1. Dyrektywa PAM	260
5.3.1.2. Dyrektywa SLA	260
5.3.1.3. Dyrektywa BIB	261
5.3.1.4. Dyrektywa TRA	261
5.3.1.5. Dyrektywa TAB	262
5.3.1.6. Dyrektywa MON	262
5.3.1.7. Dyrektywa IGN	263
5.3.2. Monity i sygnalizacja błędów.	264
5.3.3. Translator Zakończeń	265
5.3.4. Czołówka wymagana przez System Operacyjny SO 141	266
5.3.5. Przykład użycia symbolicznego wejścia i wyjścia	268
5.4. UWAGI KOŃCOWE	269
5.4.1. Sieci działań	269
5.4.2. Optymalne pisanie programów	270
5.4.3. Uruchamianie programów	272
Dodatek A LISTA SŁÓW ZASTRZEŻONYCH JĘ- ZYKA COBOL	275
Dodatek B ALFABETYCZNY SPIS ZDAŃ, INSTRU- KCJI ORAZ ZWROTÓW JĘZYKA COBOL DLA MASZYN YAM 41	279
Dodatek C TABELA NAZW ZNAKÓW KWS	281
Dodatek D ZASADY DOŁĄCZANIA MODUŁÓW DO PROGRAMU WYNIKOWEGO	283

- WYKAZ MODUŁÓW WYKORZYSTY- WANYCH W POSZCZEGÓLNYCH OPERACJACH PROGRAMU ŹRÓ- DŁOWEGO	284
- WYKAZ DŁUGOŚCI POSZCZEGÓL- NYCH MODUŁÓW	290
Dodatek E MONITY CZASU TŁUMACZENIA . .	292
Dodatek F MONITY CZASU WYKONANIA . . .	321
Dodatek G MONITY TRANSLATORA ZAKOŃCZEŃ.	337
Dodatek H PRZYKŁADY PROGRAMÓW W JĘZYKU COBOL	345

1. W S T Ą P

1.1. KRÓTKA CHARAKTERYSTYKA JĘZYKA COBOL

Nazwa języka COBOL jest skrótem angielskiego określenia Common Business Oriented Language (Uniwersalny Język do Celów Administracyjnych). W tej nazwie podane są dwie najważniejsze cechy tego języka: przeznaczenie do zastosowania w problemach administracyjnych i ekonomicznych (problemach przetwarzania danych) oraz jego uniwersalność, rozumiana jako maksymalne uniezależnienie się od maszyny.

W większości implementacji języka COBOL (czyli realizacji na konkretnej maszynie) używa się słów angielskich jako słów zastrzeżonych tego języka.

Opracowanie języka COBOL jest wynikiem wysiłków zmierzających do stworzenia takiego języka programowania, który byłby podobny do języka potocznego używanego w pracy administracyjnej. Taki język programowania byłby stosunkowo prosty do opanowania przez osoby znające problemy przetwarzania danych administracyjno-ekonomicznych, a mające mało wiadomości o maszynie.

COBOL stanowi pomost między językiem potocznym, używanym przy pracy administracyjnej i ekonomicznej, a językiem maszyny. COBOL oparty jest na języku potocznym, ale ponieważ ma być zrozumiały dla maszyny - musi być precyzyjny i trzeba w nim ściśle przestrzegać określonych reguł. Programista musi ściśle przestrzegać reguł podanych w szczegółowym opisie języka.

Źródłową publikacją dotyczącą języka COBOL jest "PROPOSED USA STANDARD COBOL" X3.4 COBOL INFORMATION BULLETIN No 9, USASI WORKING GROUP X3.4.4, COBOL STANDARDS, 1967 r. Spełniając życzenie wydawcy wyrażone we wstępie do powyższej publikacji zamieszczamy poniżej wyjątek z tego wydawnictwa.

"Każda instytucja zainteresowana przedrukiem całości lub części materiału zawartego w niniejszym opisie języka COBOL lub wykorzystaniem pomysłów opublikowanych w tym opisie do opracowania podręcznika lub w innym celu - ma prawo to uczynić. Jednak wszystkie takie instytucje są proszone o umieszczenie tego fragmentu jako części wstępu do swojej publikacji. Używający jedynie krótkich wyjątków tego opisu, np. w przeglądach bibliograficznych, nie

muszą cytować tego tekstu, są jednak proszeni o umieszczenie w tych wyjątkach hasła "COBOL".

Z języka COBOL mogą korzystać wszyscy zainteresowani, gdyż nie jest on własnością żadnej firmy, grupy firm, instytucji czy zespołu instytucji.

Żadna z instytucji, które współdziałały przy definiowaniu tego języka ani też sam COBOL COMMITTEE nie dają jednak gwarancji co do dokładności opisu i jakości funkcjonalnej języka, względnie systemu oprogramowania zawierającego ten język. W związku z tym instytucje współpracujące oraz Komitet oświadczają, że nie ponoszą żadnej odpowiedzialności za skutki wynikłe z wykorzystania języka COBOL w jakiegokolwiek formie.

Każdy może wziąć udział w pracach nad rozwojem języka COBOL. Pytania dotyczące sposobów wnoszenia zmian należy kierować do Komitetu Wykonawczego organizacji CONFERENCE ON DATA SYSTEMS LANGUAGES.

Autorami oraz właścicielami praw wydawniczych materiału opublikowanego w powyższym opracowaniu są:

FLOW-MATIC (znak firmowy SPERRY RAND CORPORATION), Oprogramowanie maszyny UNIVAC (R) I i II, Data Automation Systems, wyd. 1958, 1959, Sperry Rand Corporation;

IBM, IBM Commercial Translator, Symbol F28-8013, wyd. 1959, IBM;

FACT, DSI 27A5260-2760, wyd. 1960, Minneapolis Honeywell.

Wyżej wymienieni zezwolili na wykorzystanie tego materiału w całości lub częściowo. Zezwolenie to dotyczy również przedruku i użycia tego materiału w opisach języka COBOL i w podobnych wydawnictwach".

1.2. IMPLEMENTACJA DLA MASZYN YAM 41

W niniejszym podręczniku opisano konkretną wersję języka COBOL, której translator dla maszyny YAM 41 wykonany został w latach 1968-69 w Instytucie Maszyn Matematycznych w Warszawie.

Implementacja ta oparta jest na publikacji "PROPOSED USA STANDARD COBOL" X3.4 COBOL INFORMATION BULLETIN N° 9, USASI WORKING GROUP X3.4.4 COBOL STANDARDS, 1967 i zawiera wszystkie ważniejsze cechy języka, z wyjątkiem "report writer" oraz dostępu i przetwarzania swobodnego.

Do programu można włączać wstawki pisane w języku będącym podzbiorem języka SAS (patrz YAM 41, OPROGRAMOWANIE, Część II, JĘZYK SAS, IMM, 1969).

Do języka włączono 2 instrukcje ułatwiające uruchomienie programu: READY TRACE i RESET TRACE.

1.3. UKŁAD PODRĘCZNIKA

Podręcznik składa się ze wstępu i czterech części. Pierwsza część (oznaczona pkt 2) zawiera ogólne wiadomości o języku i może być opuszczona przez Czytelnika, znającego już inną implementację języka COBOL.

W pkt 3 omówiono sposób przedstawienia składni języka (czyli tzw. metajęzyk) stosowany w

niniejszym podręczniku, pojęcia podstawowe oraz organizację danych. Czytelnik znający już inną implementację języka COBOL może opuścić większość materiału dotyczącego pojęć podstawowych.

W punkcie 4 podręcznika omówiona jest szczegółowo składnia i semantyka języka COBOL z uwzględnieniem podziału programu na części IDENTIFICATION, ENVIRONMENT, DATA i PROCEDURE. Instrukcje występujące w PROCEDURE DIVISION zostały podzielone na następujące grupy: instrukcje związane z wejściem-wyjściem, instrukcje obliczeniowe i przesłania, instrukcje warunkowe, instrukcje związane z sortowaniem dokumentów, instrukcje sterujące translacją, instrukcje pomocnicze. Ta część podręcznika zawiera też opis języka wstawek.

W punkcie 5 podręcznika omówiono tworzenie zbiorów bibliotecznych i sposób korzystania z Biblioteki COBOL, zagadnienie ponawiania programu oraz język operacyjny COBOL-u: czołówkę, dyrektywy, monity i sygnalizację błędów.

W celu uniknięcia zbyt wielu powtórzeń, w niektórych miejscach występują odwołania do pojęć omówionych dalej. W tych przypadkach dla pełnego zrozumienia zagadnienia konieczne jest zapoznanie się z fragmentami, do których się odwołano.

Podręcznik zawiera osiem Dodatków oznaczonych literami A - H. Dodatki te zawierają materiały potrzebne w trakcie pisania programu, a więc: listę słów zastrzeżonych, spis wszystkich instrukcji i zwrotów języka COBOL, kod wewnętrzny maszyny ZAM 41, długość podprogramów włączanych do programu wynikowego, monity czasu tłumaczenia, czasu wykonania i Translatora Zakończeń. Dołączono także przykłady uruchomionych programów wraz z wynikami.

2. OGÓLNE WIA DOMO ŚCI O JĘZ YKU CO BOL

2.1. PRZEZNACZENIE I TYPOWE ZASTOSOWANIA

Jak już mówiliśmy, język COBOL został utworzony do zastosowań w pracach administracyjnych i ekonomicznych; prace te charakteryzują się dużą ilością danych i stosunkowo małym skomplikowaniem operacji.

Istotą tych prac jest odczytywanie informacji z pewnych dokumentów, wykonywanie nieskomplikowanych operacji na wartościach znalezionych w tych dokumentach i tworzenie pewnych dokumentów wyjściowych. Wielokrotne użycie słowa "dokument" w powyższym zdaniu odzwierciedla

istotną i podstawową rolę dokumentu w każdym programie COBOL.

Zastosowanie języka COBOL umożliwia pisanie programów, składających się z tysięcy rozkazów maszyny, w postaci znacznie skróconej i podobnej do języka angielskiego używanego w administracji. Program napisany w COBOL-u jest nie tylko znacznie krótszy, a więc łatwiejszy do napisania, ale jest także uniwersalny, niezależny od rodzaju maszyny. Różne maszyny mają różne zestawy podstawowych działań i różne kody na oznaczenie nawet tych samych instrukcji - dlatego program napisany w języku jednej maszyny nie będzie działał na innej maszynie; gdy zmieni się maszynę, nawet na większą i szybszą, programy napisane w języku dawnej maszyny stają się bezużyteczne. Natomiast program napisany w COBOL-u może być wykonany - po przetłumaczeniu na język maszyny za pomocą odpowiedniego translatora - prawie na każdej maszynie o wystarczającej wielkości i szybkości.

Programista używający COBOL-u nie musi uczyć się specjalnych kodów maszynowych. Wystarczy mu zaznajomienie się z dokładnym działaniem operacji nazwanych w tym języku adekwatnymi angielskimi słowami. W programie można stosować np. zwykły algebraiczny zapis działań arytmetycznych.

Można obrazowo powiedzieć, że język COBOL jest zbiorem prefabrykatów, z których programista może łatwo, szybko i bezbłędnie składać długi program wykonujący prace administracyjne.

Na ogół każde zdanie języka COBOL reprezentuje ciąg prostych instrukcji, które maszyna ma wykonać.

Program napisany w COBOL-u, tak jak zresztą program napisany w dowolnym innym języku, nie rozwiązuje żadnego problemu, którego człowiek sam nie mógłby rozwiązać. Program służy tylko do dokładnego i możliwie prostego przekazania maszynie zlecenia wykonania określonych czynności. Czynności te są zawsze pojęciowo proste i dlatego można powiedzieć, że żadna maszyna matematyczna nie wykonuje nic takiego, czego nie mógłby wykonać człowiek, gdyby miał dostatecznie dużo czasu i cierpliwości.

Instrukcje, które występują w języku maszyny, można na ogół podzielić na następujące grupy: instrukcje przesyłania informacji z jednego miejsca na inne, instrukcje arytmetyczne, instrukcje porównania dwóch wielkości, badania warunków.

Instrukcje występujące w języku COBOL są bardziej złożone i przeważnie odpowiadają wielu instrukcjom maszynowym. Typowymi dla COBOL-u instrukcjami są: instrukcja wczytania dokumentu z urządzenia zewnętrznego, instrukcja przesyłania wartości określonej pozycji jednego dokumentu do innej pozycji tego samego lub innego dokumentu, instrukcja wypisania dokumentu na zewnątrz, instrukcja sortowania zbioru dokumentów wg żądanego porządku.

Jak widać, są to instrukcje odpowiadające czynnościom wykonywanym w pracy administracyjnej. COBOL może być zastosowany do rozwiązania następujących typowych problemów:

- prowadzenie magazynu tj. operowanie dokumentami zakupu, dokumentami "wydano", dokumen-

tami remanentowymi i innymi w celu sporządzania zestawień,

- wykonanie katalogów autorskich, rzeczowych lub innych na podstawie odpowiednich "fiszek",
- wykonanie listy płac,
- wykonywanie bilansów.

2.2. PROGRAM ŹRÓDŁOWY A PROGRAM WYNIKOWY, TRANSLATOR, IMPLEMENTACJA JĘZYKA

Pisząc w podręczniku o programie będziemy mieć na myśli program napisany przez programistę, czyli tzw. program źródłowy. Program źródłowy jest niezrozumiały dla maszyny. Aby program napisany w COBOL-u mógł być wykonany - musi być zamieniony na odpowiedni ciąg rozkazów maszyny. Ciąg rozkazów maszyny, odpowiadający danemu programowi źródłowemu COBOL-u, nazywamy programem wynikowym. Zamianę czyli tłumaczenie (lub translację) programu źródłowego na program wynikowy wykonuje program zwany translatorem COBOL-u.

Translator wczytuje program źródłowy, rozpoznaje poszczególne instrukcje COBOL-u, planuje pamięć operacyjną i bębnową dla programu wynikowego, przydziela urządzenia zewnętrzne, zamienia poszczególne instrukcje na odpowiednie ciągi rozkazów maszyny i łączy to wszystko w jedną całość (czyli kompiluje).

Podkreślimy tu jeszcze, że temu samemu programowi źródłowemu odpowiadają na różnych maszynach różne programy wynikowe. Zresztą na tę samą maszynę, zmieniając translator COBOL, można

otrzymać różne programy wynikowe. Program wynikowy musi jednak być taki, aby wykonywał dokładnie te czynności, które są zapisane w programie źródłowym.

Translator języka COBOL może być przewidziany na mniej lub bardziej obszerną wersję języka. Zależy to od rodzaju i wielkości pamięci maszyny, szybkości maszyny, urządzeń zewnętrznych, a także np. od czasu na wykonanie translatora.

Twórcy COBOL-u wyznaczyli nawet minimalną wersję COBOL-u, która musi być zrealizowana na to, by język był jeszcze COBOL-em.

Poszczególne wersje COBOL-u, posiadające własne translatory na określone maszyny, noszą nazwę implementacji tego języka.

Prawdą jest, że trudno o dwie dokładnie identyczne implementacje COBOL-u, ale różne implementacje mogą być tak bardzo zbliżone, że można mówić o uniwersalności tego języka.

2.3. OGÓLNA STRUKTURA PROGRAMU COBOL

Każdy program napisany w COBOL-u musi składać się z czterech części o ustalonych nazwach:

- IDENTIFICATION DIVISION
- ENVIRONMENT DIVISION
- DATA DIVISION
- PROCEDURE DIVISION.

Kolejność wystąpienia tych części w programie musi być taka jak w powyższym wyliczeniu.

W tym miejscu chcemy opisać najogólniejsze cechy każdej części programu. Musimy przy tym użyć takich pojęć jak zbiór, dokument, pozycja,

urządzenia zewnętrzne maszyny. Pojęcia te będą szczegółowo omawiane dalej. Wydaje się jednak, że znaczenie tych pojęć jest dla wszystkich dostatecznie jasne na to, aby poniższe uwagi były zrozumiałe.

IDENTIFICATION DIVISION zawiera tytuł programu; inne informacje umieszczone przez programistę nie mają wpływu na program wynikowy.

ENVIRONMENT DIVISION ze wszystkich części programu jest najbardziej związana z rodzajem maszyny. Treścią części ENVIRONMENT jest opisanie maszyny, na której program ma być wykonywany i ustalenie koniecznych powiązań między maszyną, a opisem danych tego programu.

Między innymi w ENVIRONMENT DIVISION określa się, z którymi konkretnymi urządzeniami zewnętrznymi (stacja taśmy magnetycznej, czytnik kart, drukarka wierszowa) będą związane poszczególne zbiory danych.

W DATA DIVISION określa się postać danych, z których się korzysta lub które tworzy się w danym programie. M.in. w DATA DIVISION opisuje się dokładnie wszystkie typy dokumentów występujących w danym programie. Opisanie dokumentu polega na określeniu jego struktury przez nadanie odpowiednich numerów poziomów jego pozycjom, na nadaniu nazw wszystkim jego pozycjom i na podaniu warunków, jakie muszą spełnić wartości tych pozycji.

W PROCEDURE DIVISION opisuje się czynności, jakie mają być wykonane przez program na danych opisanych w DATA DIVISION, a także pewne czynności, które ma wykonać translator.

Struktura ENVIRONMENT DIVISION i DATA DIVISION jest bardzo sztywna. Części te dzielą się na sekcje, których nazwy są ustalone. Programista może opisać każdy rodzaj danych i jego powiązanie z maszyną za pomocą niewielu zdań o z góry ustalonych schematach.

W PROCEDURE DIVISION programista ma znacznie większą swobodę. Schematy zdań są tak samo sztywne jak w innych częściach programu, ale zdania te można w najrozmaitszy sposób ze sobą wiązać tak, aby opisać czynności w każdym programie przecież inne. Część PROCEDURE można dzielić na sekcje o dowolnych nazwach, a sekcje - na paragrafy.

Ścisłe rozdzielenie w programie COBOL opisu czynności, jakie mają być wykonane przez ten program (PROCEDURE DIVISION), charakterystyki danych, na których te czynności mają być wykonane (DATA DIVISION) i opisu maszyny jest zaletą języka, gdyż upraszcza nanoszenie ewentualnych zmian do programu, a cały program jest bardziej przejrzysty. Elastyczność języka programowania (w sensie łatwości modyfikowania programu) jest jedną z najpotrzebniejszych cech języka. Modyfikowanie to może być związane ze zmianą założeń programu (wtedy zmiany w części PROCEDURE i DATA) lub ze zmianą wyposażenia maszyny (wtedy zmiany w ENVIRONMENT i ewentualnie w DATA).

Samo programowanie też jest łatwiejsze, gdy osobno opisuje się postać danych, a osobno czynności. Taka budowa programu jest dogodna także dla tłumacza.

2.4. KONCEPCJA ORGANIZACJI DANYCH W COBOL-U

Urzędnik, który ma prowadzić określoną dokumentację, musi się nauczyć pobierania pewnych danych z dokumentów przychodzących, wykonywania pewnych obliczeń i tworzenia zestawień o określonej treści i formie.

Kolejne czynności wykonywane przez takiego urzędnika mają swoje odpowiedniki w zdaniach zawartych w PROCEDURE DIVISION programu, który ma zastąpić jego pracę.

Urzędnik musi mieć także informacje o samej budowie dokumentów oraz ich rozmieszczeniu w rozmaitych zbiorach, teczkach, szufladach itd. Podobnych informacji należy dostarczyć translatorowi języka COBOL. Informacje dotyczące organizacji i formatu danych podaje się w DATA DIVISION.

W DATA DIVISION opisuje się więc hierarchię danych, długość każdej pozycji, położenie kropki dziesiętnej, wystąpienie znaku waluty. Opisuje się zarówno dokumenty dostarczane programowi (dane wejściowe), jak i dokumenty tworzone w wyniku działania programu (dane wyjściowe).

Te same dane można grupować w rozmaity sposób. Każdy ciąg danych można traktować jak pojedynczą pozycję. Np. słownik może być traktowany albo jako pojedyncza pozycja (w bibliotece), albo jako ciąg rozdziałów (czyli seria pozycji), albo jako ciąg definicji, albo jako ciąg słów, albo jako ciąg znaków drukarskich. W COBOL-u też te same dane można rozmaicie grupować, jednak trzeba przy tym uwzględniać przyjęte ograniczenia.

Podstawową jednostką danych w COBOL-u jest dokument. Dokument składa się z pozycji. Dokumenty tego samego typu łączy się w zbiory. Pozycje mogą być elementarne lub grupowe. Pozycja grupowa składa się z pozycji elementarnych i/lub grupowych. Pozycja elementarna jest to taka część informacji (zawarta w dokumencie), która w żadnym miejscu programu nie będzie dzielona. Np. pozycja o nazwie ADRES może być pozycją elementarną, gdy odwołujemy się do niej jako do ADRES. Jeśli jednak w programie chcemy mieć odwołanie do nazwy MIEJSCOWOSC, to musimy pozycję ADRES rozbić na pozycje: MIEJSCOWOSC, ULICA i NR..DOMU. Te ostatnie pozycje staną się wówczas pozycjami elementarnymi, a pozycja ADRES stanie się pozycją grupową. Pozycja grupowa może być z kolei częścią innej pozycji grupowej. Np. w dokumencie może występować pozycja grupowa o nazwie PERSONALIA, która składa się z pozycji IMIE, NAZWISKO, ADRES, DATA..URODZENIA.

Hierarchię pozycji w dokumencie opisuje się w programie za pomocą numerów poziomów.

Dokument składa się z dowolnej skończonej liczby pozycji elementarnych. Łączenie pozycji elementarnych w pozycje grupowe ma znaczenie tylko pomocnicze. Np. w programie wykonującym listę płac dokumentem może być KARTA..PRACOWNIKA składająca się z pozycji elementarnych: PIERWSZE..IMIE, DRUGIE..IMIE, NAZWISKO, WYSOKOSC..POBOROW, DZIEŃ, MIESIĄC, ROK (rozpoczęcia pracy), ZWIĄZKOWE, SPECJALNE, PODATEK, ZDROWOTNY, RODZINNY.

Pozycje elementarne PIERWSZE..IMIE, DRUGIE..IMIE i NAZWISKO mogą składać się na pozycję grupową np. o nazwie PRACOWNIK. Pozycje DZIEŃ, MIESIĄC i ROK mogą składać się na pozycję grupową np. o nazwie DATA..ROZP..PRACY. Pozycje ZWIĄZKOWE i SPECJALNE mogą tworzyć razem pozycję grupową o nazwie SKŁADKI, a pozycja grupowa SKŁADKI i elementarna PODATEK mogą tworzyć razem pozycję POTRACENIA. Analogicznie można pozycje ZDROWOTNY i RODZINNY połączyć w pozycję grupową DODATKI.

Ważne jest to, że np. wartością pozycji PIERWSZE..IMIE może być LUDWIK, pozycji DRUGIE..

IMIE - JERZY pozycji NAZWISKO - KOWALSKI a pozycja grupowa PRACOWNIK nie ma swojej wartości innej niż połączenia wartości trzech zawartych w niej pozycji elementarnych.

Najważniejszą cechą charakteryzującą dokument w programie COBOL-u jest to, że jest to porcja danych, która jest udostępniana przez jednorazowe wykonanie instrukcji READ lub WRITE. W języku COBOL nie ma możliwości pobrania (wczytania) ze zbioru lub napisania (posłania do zbioru) części dokumentu.

Zbiór występujący w języku COBOL można porównać do szufladki bądź innego pomieszczenia, w którym są przechowywane dokumenty określonego typu.

Podkreślamy tu, że w COBOL-u zbiór może zawierać dokumenty różnych typów, np. obok dokumentów KARTA..PRACOWNIKA dokumenty ZWOLNIENIE.

Niezależnie od tego czy w danym zbiorze występują dokumenty tylko jednego rodzaju, czy też różnych rodzajów, wykonanie instrukcji READ dla tego zbioru powoduje udostępnienie kolej-

nego dokumentu z tego zbioru, a poprzednio dostępny dokument staje się niedostępny.

Jeśli chcemy mieć jednocześnie dostęp do więcej niż jednego dokumentu, musimy korzystać z dodatkowych pól roboczych.

Poza dokumentami z danymi zbiór może zawierać dokumenty metrykowe; które najczęściej występują przed i po dokumentach z danymi. Dokumenty metrykowe zbiorów w języku COBOL odpowiadają informacjom, które wypisuje się na teczkach czy szufladach zawierających określone dokumenty w nieautomatyzowanej pracy administracyjnej. Dokumenty metrykowe służą do kontroli, czy dany program przetwarza właściwy zbiór, czy szpule zawierające dokumenty zbioru są zakładane w dobrej kolejności itp.

Długość zbioru nie ma związku z pojemnością szpuli; zbiór może zajmować tylko część szpuli lub być zapisany na kilku szpulach.

Bezpośrednio związane ze zbiorem są 2 instrukcje COBOL-u: OPEN i CLOSE.

Na to, aby z danego zbioru mógł być odczytany lub zapisany na nim dokument, zbiór ten musi być "otwarty" za pomocą instrukcji OPEN. "Otwarcie" zbioru polega na ogół na sprawdzeniu jego metryki - jeśli zbiór zawiera dokumenty wchodzące do danego programu (tzw. zbiór wejściowy) lub na zapisaniu metryki - jeśli zbiór zawiera dokumenty tworzone przez dany program (tzw. zbiór wyjściowy). Po opracowaniu żądanej liczby dokumentów danego zbioru należy wykonać instrukcję CLOSE dla tego zbioru. Instrukcja CLOSE przetwarza lub zapisuje metrykę zbioru. Zbiór jest "otwarty" tylko po wykonaniu instru-

kcji OPEN, a przed wykonaniem instrukcji CLOSE dla tego zbioru. Można kolejno wykonywać instrukcje OPEN i CLOSE dla jednego zbioru. Instrukcje te można porównać do otwarcia lub zamknięcia szuflady lub segregatora z dokumentami.

Gdy programista już ustalił organizację dokumentów i zbiorów musi informacje o tej organizacji przekazać translatorowi COBOL-u. Wykonuje to pisząc odpowiednie zdania w DATA DIVISION swego programu.

W DATA DIVISION opisuje się kolejno wszystkie zbiory zawierające dokumenty wykorzystywane i tworzone przez program oraz pola robocze, które są potrzebne w czasie działania programu. Pola robocze są dla maszyny tym, czym dla urzędnika kartki papieru, na których zapisuje wyniki przejściowe lub pewne stałe wielkości.

Przykład opisu struktury dokumentu podany będzie po omówieniu numeru poziomu. Opis dokumentu występujący w DATA DIVISION zawiera także informacje o typie wartości, jakie mogą przybierać wszystkie pozycje tego dokumentu (np. do 8 cyfr dziesiętnych, do 20 znaków itd.).

2.4.1. Numer poziomu

Numery poziomu służą do dokładnego określenia struktury danych. Poziom jest pojęciem, które występuje w różnych dziedzinach, także nie mających nic wspólnego z programowaniem. Np. struktura niniejszego podręcznika jest następująca: podręcznik dzieli się na 5 punktów, a te z kolei na podpunkty oznaczane dwiema, trzema itd. cyframi. Ilość cyfr występująca w numerze podpunktu jest

tym większa im niższy jest poziom tego podpunktu w hierachii układu podręcznika.

Mówiliśmy już, że najbardziej obejmującą strukturą w COBOL-u jest zbiór, składający się z dokumentów, które z kolei składają się z pozycji grupowych lub elementarnych.

Translatorowi COBOL-u trzeba dostarczyć informacji o tym, jakie dokumenty wchodzi w skład wszystkich poszczególnych zbiorów, z jakich pozycji składa się każdy rodzaj dokumentu i które pozycje są elementarne, a które grupowe.

Rozważmy dla przykładu dokument, który m.in. zawiera adres. Możemy pozycji tej nadać nazwę ADRES. Jeżeli jednak programista chce odwołać się do pozycji zawartej wewnątrz pozycji ADRES, to musi tym pozycjom także nadać nazwy, np. MIEJSCOWOSC, ULICA, NR..DOMU.

W COBOL-u w celu opisanía takiej hierarchii pozycji przypisuje się każdej pozycji odpowiedni numer poziomu. W naszym przykładzie można pozycji ADRES nadać numer poziomu 2, a pozycjom zawartym w niej-numery poziomu 4. (Im niższy jest poziom pozycji w hierarchii dokumentu, tym większą wartość ma jej numer poziomu).

Numerы poziomu nadaje się w opisie dokumentu w DATA DIVISION. Przy nadawaniu pozycjom numerów poziomu obowiązują następujące reguły:

- 1) każdy dokument ma numer poziomu 1 (lub 01),
- 2) pozycjom w dokumencie można nadawać numery poziomów od 02 do 49 (do specjalnych celów służą numery poziomu 88 oraz 77),
- 3) na to, aby pozycja B była zawarta w innej pozycji A, muszą być spełnione następujące warunki:

- a) opis pozycji B musi występować za opisem pozycji A,
 - b) numer poziomu pozycji B musi mieć wartość większą od numeru poziomu pozycji A,
 - c) między opisem pozycji A, a opisem pozycji B nie może wystąpić ani jeden opis pozycji o numerze poziomu równym lub niższym od numeru poziomu pozycji A,
- 4) pozycja grupowa może zawierać więcej niż jeden poziom podpozycji (np. pozycja ADRES może składać się z pozycji MIEJSCOWOSC, ULICA i NR..DOMU, a pozycja MIEJSCOWOSC może składać się z pozycji elementarnych POWIAT i GROMADA). Pozycje wchodzące w skład tego samego poziomu muszą mieć ten sam numer poziomu (w naszym przykładzie jeden numer poziomu mają pozycje MIEJSCOWOSC, ULICA i NR..DOMU, a inny numer poziomu muszą mieć pozycje POWIAT i GROMADA).

Zgodnie z powyższymi regułami będziemy chcieli opisać strukturę karty pracownika, wykorzystywanej przy opracowywaniu listy płac. Karta ta ma następujące pozycje: PRACOWNIK, WYSOKOSC..POBOROW, DATA..ROZP..PRACY, POTRACENIA i DODATKI. Elementarną pozycją jest tylko pozycja WYSOKOSC..POBOROW, pozycja PRACOWNIK składa się z pozycji elementarnych IMIE..PIERWSZE, IMIE..DRUGIE i NAZWISKO. Pozycja DATA..ROZP..PRACY składa się z pozycji elementarnych DZIEŃ, MIEŚIĄC, ROK. Pozycja POTRACENIA składa się z elementarnej pozycji PODATEK i pozycji SKŁADKI składającej się z 2 elementarnych pozycji ZWIĄZKOWE i INNE. Pozycja DODATKI składa się z 2 pozycji elementarnych ZDROWOTNY i RODZINNY.

Dokument KARTA..PRACOWNIKA można naszkicować w ten sposób :

PRACOWNIK	PIERWSZE..IMIE	
	DRUGIE..IMIE	
	NAZWISKO	
WYSOKOSC..POBORÓW		
DATA..ROZP.. PRACY	DZIEŃ	
	MIESIĄC	
	ROK	
POTRACENIA	SKŁADKI	ZWIĄZKOWE
		SPECJALNE
	PODATEK	
DODATKI	ZDROWOTNY	
	RODZINNY	

Poprawnym opisem struktury takiego dokumentu będzie np.:

01 KARTA..PRACOWNIKA .

03 PRACOWNIK

05 PIERWSZE..IMIE

05 DRUGIE..IMIE

05 NAZWISKO

03 WYSOKOSC POBORÓW

03 DATA..ROZP..PRACY

08 DZIEŃ

08 MIESIĄC

08 ROK

03 POTRACENIA

05 SKŁADKI

08 ZWIĄZKOWA

08 SPECJALNE

05 PODATEK

03 DODATKI

04 ZDROWOTNY

04 RODZINNY

W powyższym opisie dokumentu dla uproszczenia opuszczono opisy własności poszczególnych pozycji; opisy te są konieczne przy pisaniu całego programu.

Najwyższemu poziomowi pozycji nadaliśmy tu numer poziomu 03 (równie dobrze mógłby być 02 lub 04). Jeśli poziomowi temu nadalibyśmy numer poziomu 04, to trzeba by koniecznie zmienić numery pozycji ZDROWOTNY i RODZINNY na dowolny inny, większy niż 04.

Ponieważ pozycji PIERWSZE..IMIE nadaliśmy numer poziomu 05, to pozycje DRUGIE..IMIE i NAZWISKO muszą mieć ten sam numer poziomu (reguła 3).

Natomiast pozycje DZIEN, MIESIAC i ROK z jednej strony, a pozycje ZWIAZKOWA i SPECJALNE z drugiej strony mają te same numery poziomu przypadkowo.

2.4.2. Specjalne numery poziomu

Oprócz numerów poziomu od 01 do 49 istnieją specjalne numery poziomu 77 i 88.

Numer poziomu 77 służy do wprowadzania pojedynczych pozycji (tzn. takich, które nie mają dalszej struktury ani nie są częściami innej struktury) w sekcji WORKING..STORAGE. Pozycje te można by też opisać jako dokumenty, które są zarazem pozycjami elementarnymi, ale wprowadzenie pozycji o numerze poziomu 77 jest prostsze.

Numer poziomu 88 służy do wprowadzania tzw. nazw warunkowych, czyli do nadawania nazw określonym zbiorom wartości pewnych pozycji. Poziom 88 piszemy bezpośrednio za opisem pozycji, której wartościom chcemy nadać nazwę warunkową.

Jeśli np. napiszemy w opisie dokumentu:

07 WIEK

88 DZIECKO;VALUE IS 0 THRU 18

88 PRODUKCYJNY; VALUE IS 19 THRU 60

88 EMERYT;VALUE IS 61 THRU 150

to będziemy mogli w programie używać zwrotów IF DZIECKO, IF PRODUKCYJNY czy IF EMERYT zamiast badać przedziały wieku.

2.4.3. Kwalifikowanie nazw

W programie można nadać tę samą nazwę różnym pozycjom np. ze względów mnemotechnicznych. Po to, aby przy każdym odwołaniu wiadomo było o jaką pozycję chodzi, nazwy, które nie są jedyne w programie - muszą być kwalifikowane za pomocą nazw swoich pozycji nadrzędnych. W przytoczonym wyżej przykładzie kwalifikowanie to wyglądałoby tak:

NAZWISKO IN KARTA..PRACY/

NAZWISKO IN ZWOLNIENIE

(Zamiast słowa IN można używać słowa OF).

To samo dotyczy nazw proceduralnych nadawanych przez programistę, czyli nazw paragrafów w PROCEDURE DIVISION. W różnych sekcjach nazwy paragrafów mogą się powtarzać, a w celu ujednozna-

cznienia odwołań do takich paragrafów kwalifikuje się je nazwami sekcji, w których są zawarte. Jeśli np. w sekcji S1 występuje paragraf P i w sekcji S2 też występuje paragraf P, to każde odwołanie np. GO TO P jest błędne, bo jest niejednoznaczne. Poprawne odwołanie musi brzmieć "GO TO P OF S1" lub "GO TO P IN S2".

Nazwę można kwalifikować więcej niż jedną nazwą pozycji obejmujących. Np. jeśli w dokumencie KARTA..PRACY występuje pozycja DATA..ROZP..PRACY składająca się z pozycji DZIEŃ, MIESIĄC i ROK i w dokumencie PRZENIESIENIE występuje pozycja DATA składająca się z pozycji DZIEŃ, MIESIĄC i ROK to odwołanie należy kwalifikować następująco:

DZIEŃ IN DATA..ROZP..PRACY IN KARTA..PRACY i
DZIEŃ OF DATA..ROZP..PRACY IN PRZENIESIENIE

lub

DZIEŃ OF KARTA..PRACY
DZIEŃ IN PRZENIESIENIE

Przy kwalifikowaniu nazw konieczne jest użycie takiej liczby kwalifikatorów (czyli nazw pozycji zawierających pozycję o nazwie, którą się kwalifikuje), aby uzyskana nazwa kwalifikowana była jednoznaczna.

Dopuszczalne jest podawanie większej liczby kwalifikatorów niż jest to konieczne dla uzyskania jednoznaczności.

3. POJĘCIA PODSTAWOWE

3.1. METAJĘZYK

Metajęzyk służy do definiowania składni (postaci syntaktycznej) języka programowania. Przyjęty w niniejszym opracowaniu metajęzyk oparty jest na notacji Backusa¹⁾. Każda jednostka syntaktyczna języka została zdefiniowana w tej notacji, nawet tam, gdzie to nie było konieczne. Formaty (definicje postaci jednostek) wszystkich jednostek zostały zgrupowane na wkładkach umieszczonych na końcu rozdziału ze względu na to, że nie mieszczą się na wąskich stronach wydawnictwa. Krótsze formaty umieszczono także w tekście.

Język programowania zbudowany jest z tzw. elementów, którymi mogą być:

1) Backus J.W.: The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference, ICIP, Paris 1959.

- pojedyncze znaki (ze zbioru znaków języka)
np.

+ A ; 7 - . ,

- ciągi znaków zwane słowami np.

ALA2 -125 'PQR = R' READ

- ciągi zbudowane ze słów oraz pojedynczych znaków np.

COMPUTE A = XY1 + 74.15 * D ;

Elementy języka rozdzielane są spacjami lub innymi znakami pełniącymi rolę separatorów.

Elementy języka mogą mieć swoje nazwy. W programie występują elementy odpowiadające tym nazwom, a nie same nazwy. Postać języka zdefiniowana jest za pomocą tzw. formatów oraz pewnych reguł. Format przedstawia budowę elementów oraz konfigurację w jakiej mogą one występować.

Formaty zbudowane są z następujących jednostek:

- z elementów języka występujących bezpośrednio,
- z nazw elementów języka,
- z symboli metajęzyka.

Elementy języka występujące bezpośrednio napisane są w formatach w takiej postaci, w jakiej mają występować w programie.

Nazwy elementów napisane są małymi literami i ujęte są w specjalne nawiasy. W programie zamiast nazw muszą występować odpowiadające im elementy.

Symbole metajęzyka nie występują w programie.

Zbiór symboli metajęzyka jest następujący:

$\langle \rangle ::= | \{ \} [] [] [] [] \text{ — } \text{ — }$

Znaczenie poszczególnych symboli:

$\langle a \rangle$	a jest nazwą elementu,
$\langle a \rangle ::=$	wszystko to, co wystąpi po prawej stronie znaku $::=$ jest elementem o nazwie a,
$A B$	w tym miejscu programu może wystąpić albo A albo B,
$\left\{ \begin{array}{l} A \\ B \end{array} \right\}$	w tym miejscu programu może wystąpić albo A albo B,
$\{A B\}$	w tym miejscu programu może wystąpić albo A albo B,
$[A]$	opcja; element A może nie wystąpić, jednak wystąpienie lub nie wystąpienie tego elementu ma istotne znaczenie,
$[A]$	lista (ciąg) elementów A; liczba elementów nie jest sprecyzowana, przynajmniej jeden element A musi wystąpić,
$[x A]$	to samo co znaki $[A]$ z tym, że elementy listy muszą być rozdzielone znakami oznaczonymi x. x reprezentuje dowolny znak języka,
 — 	podkreślenie; dotyczy tylko pewnej klasy słów (identyfikatorów będących słowami zastrzeżonymi); słowa zastrzeżone podkreślone muszą wystąpić w programie, a nie podkreślone nie muszą; wystąpienie lub nie wystąpienie słów zastrzeżonych nie podkreślonych nie ma żadnego znaczenia;

podkreślenie występuje tylko w formatach, a nie w programie; innych elementów nie podkreśla się choć z reguły muszą wystąpić.

Jeżeli grupa elementów objęta jest jedną z par nawiasów: [] [] [] to nie musi już być obejmowana nawiasami { } .

Przykłady zastosowania metajęzyka

$\langle \text{litera} \rangle ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O$
 $P|Q|R|S|T|U|V|W|X|Y|Z$

$\langle \text{cyfra} \rangle ::= 0|1|2|3|4|5|6|7|8|9$

Definicje te oznaczają, że znaki A, B, C ... Z itd. nazywają się literami, a znaki 0,1,2...9 nazywają się cyframi.

$\langle \text{liczba całkowita} \rangle ::= [+ | -] [\langle \text{cyfra} \rangle]$

Format ten wskazuje, że liczba całkowita jest ciągiem cyfr, który może, ale nie musi, zaczynać się znakiem + lub - , np.

+125 -7 30

$\langle \text{identyfikator} \rangle ::= \langle \text{litera} \rangle [[[\langle \text{litera} \rangle] [\langle \text{cyfra} \rangle]]]$

Z tego widać, że identyfikatorem może być jedna litera lub ciąg, który musi zaczynać się od litery. Następnymi znakami mogą być litery, cyfry lub dwie kropki. Dwie kropki nie mogą wystąpić na końcu np.

A1 BC..3 XB..D..KR5 Y Z

$\langle \text{zwrot DATA} \rangle ::= \text{DATA RECORDS ARE}$

$[[\langle \text{identyfikator} \rangle]]$

Format ten wskazuje, że zwrot DATA składa się z dwu słów, które muszą wystąpić DATA, RECORDS,

słowa ARE, którego można nie pisać oraz z ciągu identyfikatorów rozdzielonych przecinkami np.

DATA RECORDS ARE A, B1, XC..3

DATA RECORDS ZOSIA, JAS, RKS

3.2. DEFINICJE POJĘĆ PODSTAWOWYCH

3.2.1. POJĘCIA PODSTAWOWE DOTYCZĄCE PROGRAMU

3.2.1.1. Zbiór znaków języka

Zbiór znaków języka zawiera wszystkie znaki, które mogą wystąpić w programie. Znaki języka dzielą się na litery, cyfry i znaki specjalne.

<litera> ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q
R|S|T|U|V|W|X|Y|Z

<cyfra> ::= 0|1|2|3|4|5|6|7|8|9

<znak specjalny> ::= + | - | * | / | , | = | (|) | . | ; | ' |

Nazwy poszczególnych znaków specjalnych podaje poniższa tabelka

<u>nazwa</u>	<u>postać graficzna</u>
plus	+
minus	-
gwiazdka	*
znak dzielenia	/
przecinek	,
znak równości	=
nawias otwierający	(
nawias zamykający	)
kropka	.
średnik	;
apostrof	'
spacja	

dnej zmiennej warunkowej, to w odwołaniach nazwę tę trzeba kwalifikować w celu uzyskania jednoznaczności. Kwalifikatorem nazwy warunku może być jej zmienna warunkowa albo którakolwiek pozycja nadrzędna względem tej zmiennej.

3.2.1.2.1.3. N a z w a w s k a ź n i k a

Nazwa wskaźnika jest nazwą określonego bitu trzeciego słowa maszyny ZAM 41. Nazwa wskaźnika musi w programie być jednoznaczna gdyż nie można jej kwalifikować.

3.2.1.2.1.4. N a z w a p r o c e d u r y

Nazwa procedury jest to nazwa sekcji lub nazwa paragrafu w PROCEDURE DIVISION.

W jednym programie nie może być dwóch sekcji o tej samej nazwie, a w jednej sekcji dwóch paragrafów o tej samej nazwie. Jeśli w różnych sekcjach są paragrafy o takiej samej nazwie - to w odwołaniach trzeba tę nazwę paragrafu kwalifikować za pomocą nazwy sekcji.

Zaznaczamy, że nazwy sekcji i paragrafów występujące poza PROCEDURE DIVISION są słowami zastrzeżonymi i nie są nazwami procedur.

3.2.1.2.1.5. S ł o w o z a s t r z e ż o n e

Słowa zastrzeżone są podstawowymi elementami składni języka. Słowa zastrzeżone są odpowiednikami kodów instrukcji lub dyrektyw w języku maszyny. Odróżniamy słowa zastrzeżone kluczowe i opcjonalne.

Słowo zastrzeżone kluczowe określa typ i funkcję jednostki syntaktycznej, w której występuje; wystąpienie lub brak określonego słowa kluczowego w jednostce syntaktycznej zmienia treść tej jednostki. Do słów zastrzeżonych kluczowych należą wszystkie słowa, które w formatach poszczególnych jednostek syntaktycznych są podkreślone (w tym również wskaźniki FD oraz SD).

Słowa zastrzeżone opcjonalne występują w formatach poszczególnych jednostek syntaktycznych w celu zwiększenia ich czytelności w języku angielskim. Słowa te mogą występować w określonych miejscach programu, lecz ich brak nie zmienia sensu programu. Słowami zastrzeżonymi opcjonalnymi są wszystkie słowa nie podkreślone w formatach poszczególnych jednostek syntaktycznych.

Słowa zastrzeżone oznaczają w programie tylko siebie, tj. nie mogą być używane jako nazwy wielkości definiowanych przez programistę, np. jako nazwy jednostek danych albo nazwy procedur.

Lista słów zastrzeżonych znajduje się w Dodatku A.

3.2.1.2.1.6. S t a ł a f i g u r a t y w n a

Stała figuratywna jest słowem zastrzeżonym będącym nazwą liczby zero lub określonego tekstu. Odróżniamy następujące stałe figuratywne:

ZERO	oznacza wartość liczbową zero;
	jest wielkością obliczeniową typu FIXED,

SPACE	oznacza jeden lub więcej znaków SP (patrz kod KW8); jest pozycją znakową,
LOW..VALUE	oznacza jeden lub więcej znaków NU (patrz kod KW8); jest pozycją znakową,
HIGH..VALUE	oznacza jeden lub więcej znaków DE (patrz kod KW8); jest pozycją znakową.

Stała figuratywna ZERO może występować w programie wszędzie tam, gdzie mogą występować nazwy pozycji typu FIXED.

Stałe figuratywne SPACE, LOW..VALUE i HIGH..VALUE mogą występować w programie wszędzie tam, gdzie mogą występować nazwy pozycji typu znakowego.

Długość tekstu będącego wartością stałej figuratywnej SPACE, LOW..VALUE i HIGH..VALUE określana jest w czasie translacji na podstawie kontekstu, w którym określona stała figuratywna występuje:

- a) gdy stała figuratywna jest przesyłana do jakiejś jednostki danych lub porównywana z taką jednostką, długość stałej figuratywnej czyni się równą długości tej jednostki;
- b) gdy stała figuratywna występuje w instrukcji DISPLAY, EXAMINE lub STOP jest ona tekstem składającym się z jednego znaku.

3.2.1.2.1.7. Nazwy specjalnych rejestrów TALLY i DRUM

Nazwą specjalnego rejestru jest słowo TALLY i słowo DRUM.

W rejestrze TALLY przechowywana jest informacja przesłana w wyniku wykonania instrukcji EXAMINE oraz READ.

Nazwa TALLY może występować w programie na miejscu nazw jednostek typu FIXED.

Drugim rejestrem specjalnym jest rejestr DRUM. Rejestr ten zawiera adres początku wolnego obszaru na bębnie, czyli adres, począwszy od którego bęben jest do dyspozycji programisty i ewentualnie akcji RERUN, natomiast początkowa część bębna jest zajęta przez program wynikowy.

3.2.1.2.1.8. Nazwa pozycji metrykowej

Nazwa pozycji metrykowej jest to słowo zastrzeżone będące nazwą pozycji metryki początku zbioru (zob. punkt 3.3.3.1.4.1). Nazwami pozycji metrykowych w tej implementacji COBOL-u są: SECT, IDENT, CREAT, GENER oraz EXPIR. Ponadto do pozycji metrykowych zaliczamy również SECUR, które nie należy do żadnej metryki, ale ma wartość

$$\text{SECUR} = \text{EXPIR} - \text{CREAT}$$

Do nazw pozycji metrykowych wolno odwoływać się jedynie w ramach instrukcji USE określających niestandardowe procedury metrykowe. Znaczenie poszczególnych pozycji metrykowych podane jest w pkt 3.3.3.

3.2.1.2.1.9. Nazwa zbioru

Nazwa zbioru jest to identyfikator, za pomocą którego opisuje się w DATA DIVISION zbiór danych lub zbiór sortowania. Nazwa zbioru deklarowana jest w DATA DIVISION po wskaźniku FD lub SD.

Zmienna BUFF jest przeznaczona do wypisywania zawartości bufora zbioru, szczególnie w przypadku błędu w sekcjach USE (porównaj opis zbioru niepozycyjnego oraz zdania USE).

Zmienna **HARDWARE**

Zmienną **HARDWARE** wprowadzono w celu umożliwienia programiście zmiany urządzenia przydzielonego zbiorowi danych - zmiany w czasie działania programu. Jest to zmienna typu **FIXED**, której wartością jest numer urządzenia ostatnio przydzielonego zbiorowi. Ponieważ dla zbioru sortowania tworzy się n zbiorów pomocniczych (gdzie n jest równa ilości urządzeń przydzielonych zbiorowi sortowania), to numery tych urządzeń zawierają zmienne o nazwach **HARDWAREi**, gdzie i może przyjmować wartości od 1 do n .

Zmienna **HARDWARE** i **HARDWAREi** powinna być kwalifikowana nazwą zbioru, jeśli w programie występuje więcej niż jeden zbiór danych i/lub zbiór sortowania (porównaj opis deklaracji **SELECT**).

3.2.1.2.2. Stała

Stała może być:

liczbą,
tekstem,
numerem poziomym,
obrazem.

$\langle \text{stała} \rangle ::= \langle \text{liczba} \rangle | \langle \text{tekst} \rangle | \langle \text{numer poziomym} \rangle |$
 $\quad \quad \quad \langle \text{obraz} \rangle$

3.2.1.2.2.1. L i c z b a

Liczba jest wielkością, która w trakcie tłumaczenia programu przekształcana jest na postać binarną. Rozróżniamy trzy rodzaje liczb: całkowite, całkowite dwuprecyzyjne i zmiennie-przecinkowe.

$\langle \text{liczba} \rangle ::= \langle \text{liczba całkowita} \rangle | \langle \text{liczba całkowita dwuprecyzyjna} \rangle | \langle \text{liczba zmiennie-przecinkowa} \rangle$

Liczba całkowita

Liczba całkowita jest ciągiem co najwyżej siedmiu cyfr, ewentualnie poprzedzonym znakiem plus lub minus. Największą dopuszczalną bezwzględną wartością liczby całkowitej jest 8 388 607, tj. $2^{23}-1$. Jeżeli w programie występuje liczba całkowita większa co do wartości bezwzględnej od $2^{23}-1$, to jest ona automatycznie traktowana jak liczba całkowita dwuprecyzyjna.

$\langle \text{liczba całkowita} \rangle ::= [\pm] \llbracket \langle \text{cyfra} \rangle \rrbracket$

Liczba całkowita dwuprecyzyjna

Liczba całkowita dwuprecyzyjna jest ciągiem co najmniej siedmiu i co najwyżej czternastu cyfr, ewentualnie poprzedzonym znakiem plus lub minus. Największą dopuszczalną wartością bezwzględną liczby całkowitej dwuprecyzyjnej jest 70 368 744 177 663, tj. $2^{46}-1$.

$\langle \text{liczba całkowita dwuprecyzyjna} \rangle ::=$
 $\quad \quad \quad [\pm] \llbracket \langle \text{cyfra} \rangle \rrbracket$

Liczba zmiennoprzecinkowa

W celu zdefiniowania liczby zmiennoprzecinkowej podamy pomocnicze definicje cechy i liczby ułamkowej.

Cecha jest to ciąg złożony z litery E, opcjonalnego znaku plus albo minus i co najwyżej dwucyfrowej liczby całkowitej, zbudowany według następujących reguł:

- a) pierwszym znakiem ciągu jest litera E,
- b) jeżeli w ciągu występuje znak plus albo minus, to zapisuje się go bezpośrednio za literą E,
- c) liczba całkowita musi należeć do przedziału domkniętego $[-77, +76]$,
- d) wewnątrz ciągu nie występuje znak spacji.

Przykłady cech:

E+12

E21

E-6

$\langle \text{cecha} \rangle ::= E \left[\begin{smallmatrix} + \\ - \end{smallmatrix} \right] \langle \text{cyfra} \rangle \left[\langle \text{cyfra} \rangle \right]$

Liczbą ułamkową nazywamy ciąg złożony z opcjonalnego znaku plus lub minus, kropki oraz cyfr, zbudowany według następujących reguł:

- a) jeżeli w ciągu występuje znak plus albo minus, to jest on pierwszym znakiem ciągu,
- b) kropka występuje w ciągu tylko raz i nie może być ostatnim znakiem ciągu,
- c) wewnątrz ciągu nie występuje znak spacji.

Przykłady liczb ułamkowych:

12.345
-62,1
+.256
.5

Liczbą zmiennoprzecinkową jest: liczba całkowita z cechą, liczba całkowita dwuprecyzyjna z cechą, liczba ułamkowa, liczba ułamkowa z cechą.

Przykłady liczb zmiennoprzecinkowych:

12E-12
-.625
+12.345E76

Mantysą nazywa się część liczby zmiennoprzecinkowej zapisaną przed literą E.

Miedzy mantysą liczby a jej cechą nie może występować spacja.

3.2.1.2.2.2. T e k s t

W celu zdefiniowania tekstu podamy pomocnicze definicje elementów tekstu.

Znakiem podstawowym będziemy nazywali dowolny znak kodu KW8 z wyjątkiem kropki, znaku dzielenia i apostrofu.

Nazwa znaku jest to mnemotechniczny skrót angielskiego terminu oznaczającego ten znak, przy czym skrót ten złożony jest z dwóch znaków podstawowych, z których pierwszy jest literą. Lista nazw znaków kodu KW8 podana jest w Dodatku C.

Symbol specjalny jest to ciąg zaczynający się od kropki (po której może następować liczba całkowita bez znaku) zakończony nazwą znaku.

$\langle \text{symbol specjalny} \rangle ::= .[\langle \text{cyfra} \rangle] \langle \text{nazwa znaku} \rangle$

Przykłady symboli specjalnych:

.LT .12OSP .YX

Przy tłumaczeniu programu napisanego w COBOL-u każdy symbol specjalny przekodowywany jest na znak, którego nazwa występuje w tym symbolu specjalnym. Jeżeli w symbolu specjalnym występuje liczba całkowita, to oznacza ona ile razy w przetłumaczonym tekście ma być powtórzony znak określony tym symbolem specjalnym. Na przykład symbol specjalny .12OSP oznacza, że na miejsce, w którym ten symbol wystąpił, należy do tekstu wstawić 120 spacji; symbol specjalny .ORP oznacza, że na jego miejsce należy do tekstu wstawić zero nawiasów zamykających. Brak liczby w symbolu specjalnym równoważny jest wystąpieniu w nim liczby '1' (jedności).

Pochodna znaku jest to ciąg zaczynający się od znaku podstawowego lub symbolu specjalnego zakończony znakiem dzielenia. W czasie tłumaczenia programu pochodna danego znaku zostaje zamieniona przez znak, który ma wartość kodową o 1 większą od wartości kodowej danego znaku.

Na przykład ciąg S/ zostanie przetłumaczony na Ś, ponieważ w KW8 mamy S=166, zaś S/=167=Ś;

.4YA/ zostanie przetłumaczony na aaaa.

$\langle \text{pochodna znaku} \rangle ::= \left\{ \begin{array}{l} \langle \text{znak podstawowy} \rangle \\ \langle \text{symbol specjalny} \rangle \end{array} \right\} /$

Elementem tekstu będziemy nazywali znak podstawowy, symbol specjalny lub pochodną znaku.

$\langle \text{element tekstu} \rangle ::= \langle \text{znak podstawowy} \rangle |$
 $\langle \text{symbol specjalny} \rangle | \langle \text{pochodna znaku} \rangle$

Tekst jest to niepusty ciąg elementów tekstu, zamknięty w apostrofach.

`<tekst>::= '[[<element tekstu>]]'`

Na przykład ciąg znaków:

`'22.SPLIPCA.SP.YM.YA.YM.IY.SPS/WIE/TO.
SPNARODOWE'`

znaczy po przetłumaczeniu:

22 LIPCA mamy ŚWIĘTO NARODOWE

Uwaga 1.

Każdy znak podstawowy, znak oznaczony symbolem specjalnym lub pochodną znaku, zajmuje w maszynie jedno słowo i zapisywany jest w kodzie KW8 na ostatnich ośmiu bitach tego słowa.

Uwaga 2.

W tekście może występować zarówno sam znak (mówimy wtedy, że znak występuje w tekście jawnie), jak i odpowiadający mu symbol specjalny¹⁾, przy czym obie te wielkości traktowane są jednakowo. Niemniej istnieje kilka znaków, których jawne wystąpienie w tekście traktowane jest specjalnie i dlatego należy je zapisywać wyłącznie za pomocą ich nazw. Są to następujące znaki: apostrof (AP), znak dzielenia (SL), kropka (FP), spacja (SP), powrót karetki (CR) oraz nowa linia (LF). Apostrof występujący jawnie kończy tekst.

1) Oczywiście jeżeli w klawiaturze określonego urządzenia zewnętrznego nie występuje jakiś znak, to programista będzie używał tylko odpowiedniego symbolu specjalnego.

Znak dzielenia występujący jawnie służy jedynie do tworzenia pochodnej znaku.

Kropka występująca jawnie oznacza początek symbolu specjalnego. Spacja, powrót karetki oraz nowa linia występujące w tekście jawnie są pomijane.

3.2.1.2.2.3. Numer poziomy

Numer poziomy jest wielkością złożoną z jednej lub dwu cyfr, przyjmującą wartości od 1 (lub 01) do 49, 77 lub 88. Używa się go w DATA DIVISION, gdzie służy do określania hierarchii danych (pkt 3.3.1). Numer poziomy opisano dokładnie w pkt 3.3.1.

3.2.1.2.2.4. Obraz

Obraz jest to zamknięty w apostrofach ciąg co najwyżej trzydziestu znaków spośród znaków podanych w Tabeli Dopuszczalnych Kontekstów Znaków w Obrazach zamieszczonej w pkt 4.3.1.8.4.

$\langle \text{obraz} \rangle ::= '[[\langle \text{symbol obrazu} \rangle]]'$

3.2.1.2.3. Punkt ponawiania programu

Punkt ponawiania programu jest to miejsce w programie, wyznaczone przez dane lub urządzenia zewnętrzne, w którym przerywana jest normalna praca programu i aktualny stan całej maszyny zapisywany jest na specjalnie do tego celu przeznaczonej taśmie magnetycznej. Wykonanie programu można wznowiać od dowolnego punktu ponawiania wykorzystując tzw. translator RESTART.

3.2.2. POJĘCIA PODSTAWOWE DOTYCZĄCE DANYCH

3.2.2.1. Zbiór znaków dla danych

Zbiorem znaków dla danych jest KW8 (zob. Dodatek C). Wszystkie omawiane w dalszych rozdziałach tego opisu operacje języka COBOL wykonywane są na danych zapisanych w maszynie w postaci binarnej albo w kodzie KW8.

W związku z tym, w przypadku wykonywania programu wynikowego, trzeba dla Symbolicznych Wejść-Wyjść określać dekodery ośmiobitowe (patrz ZAM 41, OPROGRAMOWANIE (SYSTEM 141), TOM 1, MASZYNA UŻYTKOWA).

Uporządkowanie znaków dla danych jest takie jak w KW8.

3.2.2.2. Zbiór i dokument

Zbiór dokumentów z danymi i/lub dokumentów metrykowych, zorganizowany do użycia w procesie przetwarzania danych, będziemy nazywali zbiorem danych lub krócej zbiorem.

Dokument jest to jedna lub więcej pozycji zgrupowanych w sposób dogodny dla operacji wejścia-wyjścia.

Zbiór danych jest zawsze tworzony w związku z konkretnym zagadnieniem praktycznym (np. zbiór danych opisujący aktualny stan magazynu albo zawierający listę płac), ale może być wykorzystywany w wielu programach. Dane stanowiące zbiór muszą być związane z jednym nośnikiem, np. z taśmą perforowaną albo taśmą magnetyczną, albo z kartami perforowanymi. Gdy zbiór jest związany z taśmą magnetyczną, wów-

czas może on być zapisany na kilku szpulach taśmy oraz może być związany z kilkoma stacjami taśmy.

W programie napisanym w języku COBOL określa się wszystkie własności zbioru, a więc zarówno własności, które będziemy nazywali fizycznymi tj. dotyczące postaci danych stanowiących zbiór na nośniku, jak i własności, które będziemy nazywali logicznymi, tzn. określające strukturę i typ jednostek informacji wchodzących w skład zbioru.

3.2.2.2.1. Własności fizyczne zbioru

Należą do nich własności określające:

- a) urządzenie zewnętrzne i nośnik informacji związane ze zbiorem,
- b) sposób zapisu danych na nośniku (pozycyjny lub niepozycyjny); zbiór jest pozycyjny jeżeli każda jego pozycja ma na nośniku dokładnie tyle znaków, ile wynika z jej opisu; zbiór jest niepozycyjny jeżeli pozycje jego dokumentów są na nośniku oddzielane spacją lub innym zadeklarowanym przez programistę znakiem,
- c) ograniczniki oddzielające i identyfikujące poszczególne jednostki danych w ciągu znaków na nośniku,
- d) w przypadku zbiorów związanych z taśmą magnetyczną - sposób grupowania dokumentów w bloki i sposób zapisu bloku na taśmie,
- e) punkty ponawiania programu,
- f) dane umożliwiające identyfikację zbioru,

- g) kierunek przesyłania zbioru - wejściowy, gdy dane zbioru są wczytywane do maszyny; i wyjściowy - gdy dane są wypisywane z maszyny do zbioru.

3.2.2.2.2. Własności logiczne zbioru

Należą do nich własności określające:

- a) typy dokumentów wchodzących w skład zbioru,
- b) strukturę dokumentu każdego typu i własności poszczególnych jednostek danych wchodzących do dokumentu.

3.2.2.2.3. Hierarchia danych w zbiorze

Jak wynika z definicji, każdy zbiór składa się z dokumentów. W przypadku zbiorów związanych z taśmą magnetyczną wprowadza się jeszcze dodatkowo pojęcie bloku. Z kolei każdy dokument składa się z jednostek danych nazywanych pozycjami. Pozycje, które zawierają w sobie inne pozycje będziemy nazywali grupowymi, zaś pozycje, które nie zawierają w sobie innych pozycji - elementarnymi. Organizacja danych opisana jest w pkt. 3.3.

3.2.3. PROGRAM ŹRÓDŁOWY I PROGRAM WYNIKOWY

Program źródłowy jest to program napisany w języku COBOL.

Należy tu podkreślić, że w implementacji dla maszyny ZAM 41 zapis programu źródłowego jest niepozycyjny, to znaczy, że układ graficzny programu na tabulogramie może być dowolny (z zachowaniem reguł języka). Elementy języka muszą być rozdzielane separatorami. Dodatkowe zna-

ki spacji i nowej linii mogą występować dowolnie między elementami języka.

Program źródłowy tłumaczony jest przez translator języka COBOL na ciąg rozkazów maszyny ZAM 41. Ponadto translator rezerwuje pola robocze dla danych zwane polami zbiorów.

Utworzony przez translator ciąg rozkazów wraz z polami zbiorów tworzy program wynikowy.

Pole zbioru składa się z pola dokumentu i bufora zbioru.

3.2.3.1. Pole dokumentu

Każdemu zbiorowi danych opisanemu w sekcji zbiorów w DATA DIVISION translator COBOL przydziela obszar pamięci operacyjnej, w którym przechowywane są kolejno dokumenty tego zbioru. Omawiany obszar pamięci nazywa się polem dokumentu.

Każda pozycja rozmieszczona jest w polu dokumentu zgodnie ze swoim opisem. Długość dokumentu jest zatem równa sumie pól poszczególnych pozycji wchodzących w skład dokumentu.

W jednym zbiorze danych mogą występować dokumenty różnych typów, a więc na ogół różnej długości; pole dokumentu przydzielane jest zbiorowi tak, aby w tym polu mieścił się najdłuższy spośród dokumentów zbioru.

3.2.3.2. Bufor

Niepozycyjnemu zbiorowi danych translator COBOL przydziela obszar pamięci służący do przechowywania informacji na jej drodze między nośnikiem

zewnątrznym, a polem dokumentu (na nośniku informacja jest zapisana znakowo, a w polu dokumentu niektóre pozycje są binarne). Omawiany obszar nazywany jest buforem zbioru.

Długość bufora zbioru określona jest zwrotem BLOCK CONTAINS w opisie zbioru, a jeśli nie użyje się tego zwrotu - długość bufora jest równa długości pola dokumentu. W większości przypadków, w celu zapewnienia prawidłowego działania instrukcji READ i WRITE, konieczne jest użycie zwrotu BLOCK CONTAINS określającego bufor dłuższy niż pole dokumentu (patrz pkt. 3.3).

Gdy nośnikiem informacji są taśmy magnetyczne, porcją danych jednorazowo posyланą na taśmę lub pobieraną z taśmy jest cała zawartość bufora. Stąd wniosek, że na taśmie zawartość bufora tworzy blok.

3.3. ORGANIZACJA DANYCH

3.3.1. HIERARCHIA DANYCH

Najmniejszą jednostką danych, którą operuje się w programie jest pozycja elementarna.

Pozycja elementarna może zawierać liczbę binarną lub tekst będący ciągiem znaków.

Pozycje elementarne mogą być łączone w pozycje grupowe.

Zespół pozycji elementarnych (ewentualnie połączonych w pozycje grupowe) stanowi dokument. Z kolei grupa dokumentów stanowi zbiór. W programie może występować wiele zbiorów.

Istnieją dokumenty nie zawarte w żadnym zbiorze. Są to tak zwane dokumenty robocze.

Hierarchia tych jednostek definiowana jest w części programu DATA DIVISION. Każdej jednostce przypisane są następujące parametry:

- nazwa jednostki,
- numer poziomu,
- pewne własności.

Numer poziomu jest liczbą całkowitą, która może przyjmować jedną z wartości od 01 do 49, 77 albo 88.

Miejsce jednostki w hierarchii danych określają dwa parametry:

- wartość numeru poziomu,
- położenie opisu jednostki względem opisów innych jednostek.

Pozycja o numerze poziomu n zawarta jest w najbliższej poprzedzającej ją w opisie dokumentu pozycji o numerze q gdy $q < n$. Wówczas pozycja o numerze poziomu q zawiera pozycję o numerze n .

Pozycją elementarną jest pozycja nie zawierająca innych pozycji. Pozycją grupową jest pozycja zawierająca inne pozycje.

Pozycja, której przypisano numer poziomu 01 jest zwana dokumentem.

Pozycja, której przypisano w miejsce numeru poziomu symbol FD lub SD jest zbiorem.

Różne instrukcje części PROCEDURE, w zależności od swej treści, operują różnymi jednostkami, np.:

- operacje arytmetyczne, warunkowe dotyczą pozycji elementarnych,

- przesłania odbywające się wewnątrz pamięci dotyczą pozycji grupowych, elementarnych oraz macierzy,
- operacje wejścia-wyjścia operują całym dokumentem lub zbiorem.

Przykład

W części DATA wystąpił następujący opis

⋮

FD ANKIETA; ...

01 ADRES; ...

02 ULICA; ...

02 NUMER..DOMU; ...

01 INNE; ...

02 NAZWA; ...

03 NAZWISKO; ...

03 IMIE; ...

02 DATA..URODZENIA; ...

03 ROK; ...

03 MIESIAC; ...

03 DZIEŃ; ...

SD PRZYDZIAŁY; ...

⋮

Ten fragment części DATA wskazuje, że zbiór danych ANKIETA zawiera dwa typy dokumentów: ADRES i INNE.

Dokument ADRES składa się z dwu pozycji elementarnych: ULICA i NUMER..DOMU.

Dokument INNE zawiera dwie pozycje grupowe: NAZWA oraz DATA..URODZENIA. Każda z nich skła-

da się z pozycji elementarnych. Pozycja NAZWA zawiera pozycje elementarne: NAZWISKO i IMIE, a pozycja DATA..URODZENIA zawiera pozycje: ROK, MIESIAC, DZIEŃ.

Ponadto istnieje zbiór sortowania o nazwie PRZYDZIAŁY.

3.3.2. ORGANIZACJA PAMIĘCI OPERACYJNEJ DLA DANYCH ORAZ MIEJSC ROBOCZYCH

3.3.2.1. Struktura pamięci

Danym, którymi operuje program przydziela się pewne obszary pamięci operacyjnej. Obszary te przydzielane są zbiorom lub dokumentom roboczym. Są to:

- pole dokumentu,
- bufor zbioru.

Zbiór posiada jedno pole dokumentu wspólne dla wszystkich dokumentów zbioru niezależnie od tego z ilu typów dokumentów zbiór się składa.

Natomiast każdy dokument roboczy posiada swoje oddzielne pole dokumentu.

W polu dokumentu każda pozycja elementarna ma swoje pole zwane polem pozycji. Dane przesyłane są do pola dokumentu po przekształceniu na postać wewnętrzną.

Bufor zbioru jest obszarem pomocniczym wykorzystywanym przy przesyłaniu danych między nośnikiem zewnętrznym, a polem dokumentu.

Bufor lub kilka buforów zbioru posiadają tylko te zbiory, dla których sposób przesyłania danych wymaga takiego obszaru.

Dokumenty robocze buforów nie posiadają. Zbiory sortowania posiadają pola dokumentu oraz bufor przydzielane na analogicznych zasadach jak dla zbiorów danych.

Niektóre zbiory mogą mieć wspólne z innymi zbiorami pole dokumentu i wspólny bufor.

3.3.2.2. Pole pozycji; postać danych w pamięci

Każdej pozycji elementarnej translator przypisuje pewien obszar pamięci operacyjnej (w polu dokumentu) zwany polem pozycji. Wielkość tego pola, to znaczy długość pozycji, zależy od tego, jakie własności zostały przypisane pozycji w części DATA DIVISION. I tak:

- pozycje obliczeniowe opisane zwrotem `FIXED` zajmują 1 miejsce pamięci,
- pozycje obliczeniowe opisane zwrotem `FIXED2` lub `FLOAT` zajmują dwa miejsca pamięci,
- pozycje znakowe opisane zwrotem `PICTURE` zajmują tyle miejsc pamięci, ile wystąpiło w obrazie symboli reprezentujących znaki.

Jeżeli w opisie pozycji wystąpił ponadto zwrot `OCCURS`, to pozycja zajmuje odpowiednią wielokrotność jednakowych pól pamięci, których długość określona jest jak powyżej.

Jednostki danych, które odpowiadają pozycjom elementarnym umieszczane są w pamięci operacyjnej w dwojakiej postaci:

- liczb binarnych,
- ciągów znaków (tzw. teksty).

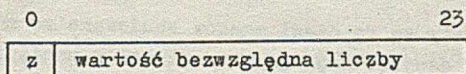
Język COBOL dopuszcza trzy rodzaje liczb:

- całkowite krótkie,
- całkowite długie,
- zmiennoprzecinkowe - znormalizowane.

Teksty są ciągami znaków ze zbioru znaków dla danych.

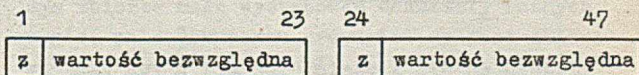
Liczba całkowita krótka zajmuje jedno słowo pamięci:

- w bicie 0 znajduje się znak algebraiczny liczby,
- w bitach pozostałych - wartość bezwzględna liczby binarnie.



Liczba całkowita długa zajmuje dwa kolejne słowa pamięci:

- w bitach 0 oraz 24 znajduje się znak algebraiczny liczby,
- w bitach pozostałych - binarna wartość bezwzględna liczby.



Liczba zmiennoprzecinkowa zajmuje dwa kolejne słowa pamięci:

- w bicie 0 znajduje się znak algebraiczny mantysy,
- w bicie 24 znajduje się znak algebraiczny cechy.

FIXED liczba całkowita krótka,
FIXED2 liczba całkowita długa,
FLOAT liczba zmiennoprzecinkowa.

Pozycje elementarne zawierające teksty nazywają się znakowymi i opisane są w części DATA zwrotem PICTURE.

3.3.2.3. Pole dokumentu

Każdemu zbiorowi danych oraz zbiorowi sortowania translator COBOL przydziela oddzielny obszar pamięci operacyjnej, w którym udostępniany jest do przetwarzania kolejny dokument tego zbioru. Nosi on nazwę pola dokumentu i obejmuje pola wszystkich pozycji wchodzących w skład dokumentu.

Każda pozycja rozmieszczona jest w polu dokumentu zgodnie ze swoim opisem w części DATA DIVISION. Długość pola dokumentu jest zatem równa sumie długości poszczególnych pozycji elementarnych wchodzących w skład dokumentu. Z wyjątkiem oczywiście pozycji opisanych zwrotem REDEFINES.

Pole to jest wspólne dla wszystkich typów dokumentów danego zbioru. W związku z tym, że w jednym zbiorze mogą występować dokumenty różnych typów, pole dokumentu przydzielane jest zbiorowi tak, aby w tym polu mieścił się najdłuższy dokument zbioru. Ogranicznik dokumentu ani pozycji nie jest umieszczany w polu dokumentu.

Jednak jeżeli w części ENVIRONMENT DIVISION wystąpi zwrot SAME, to zbiory wymienione na liście nazw zbiorów tego zwrotu otrzymają wspólne pole dokumentu. Długość tego pola jest

długości najdłuższego spośród dokumentów tych zbiorów. Przydzielanie wspólnego pola różnym zbiorom pozwala na oszczędniejsze wykorzystanie pamięci operacyjnej.

3.3.2.4. Bufor zbioru

Bufor zbioru służy do przechowywania informacji na ich drodze między nośnikiem zewnętrznym, a polem dokumentu. W buforze następuje tłumaczenie informacji z postaci zewnętrznej (na nośniku zewnętrznym) na postać wewnętrzną (w pamięci operacyjnej). Długość bufora zbioru określa się zwrotem `BLOCK` w części `DATA DIVISION`. O ile zwrot ten nie wystąpi, długość bufora zbioru danych równa jest długości pola dokumentu tego zbioru. Natomiast bufor zbioru sortowania jest obszarem pamięci o długości równej iloczynowi liczby taśm sortowania przez długość bufora określoną tak, jak dla zbioru danych. W celu lepszego wykorzystania pamięci operacyjnej kilku zbiorom można przydzielić wspólny bufor. Służy do tego zwrot `SAME AREA` w części `ENVIRONMENT`. (Zwrot ten powoduje również przydzielanie wspólnego pola dokumentu). Ponadto zbiorowi związanemu z taśmą magnetyczną można przydzielić kilka buforów, co przyspiesza opracowanie informacji. Służy do tego zwrot `RESERVE` w części `ENVIRONMENT`. Nie wszystkie zbiory danych wymagają bufora zbioru. Związane jest to z rodzajem nośnika informacji oraz sposobem czytania danych.

- a) Zbiorom pozycyjnym nie przydziela się buforu, gdyż przesłanie odbywa się bezpośrednio do pola dokumentu,

- b) zbiory niepozycyjne muszą posiadać bufor takiej długości, aby było możliwe rozpoznawanie typu wczytywanego dokumentu:
- przy technice wczytywania 1 musi mieścić się w buforze cały dokument w postaci znakowej,
 - przy technice wczytywania 2 musi mieścić się w buforze początkowa część dokumentu w postaci znakowej, zawierająca znacznik dokumentu,
 - przy technice wczytywania 3 muszą być spełnione dwa poprzednie warunki odpowiednio dla różnych sposobów rozpoznawania typu;

w buforze zbioru niepozycyjnego musi mieścić się ogranicznik zbioru oraz dokumenty specjalne; przy określaniu długości bufora zbioru niepozycyjnego należy zwrócić uwagę na to, że w buforze każda pozycja umieszczona jest znakowo (1 znak zajmuje 1 słowo pamięci); tu pozycje obliczeniowe tłumaczone są na postać binarną i w tej postaci posyłane do pola dokumentu; ogranicznik pozycji oraz dokumentu nie jest umieszczany w buforze;

- c) zbiory związane z taśmą magnetyczną muszą posiadać bufor zbioru. Porcją danych posyланą na taśmę jednorazowo jest cała zawartość bufora. Zawartość ta tworzy blok na taśmie. Porcją danych pobieranych z taśmy jest blok. O ile długość jego przekracza długość bufora to część informacji zostaje obcięta.

3.3.3. DANE NA NOŚNIKACH

3.3.3.1. Dane związane z nośnikiem papierowym

Punkt ten dotyczy danych przesyłanych instrukcjami READ oraz WRITE. Nie dotyczy natomiast instrukcji ACCEPT oraz DISPLAY.

3.3.3.1.1. Zbiór znaków dla danych

W tej implementacji jako kod wewnętrzny dla danych znakowych przyjęto ośmiobitowy kod o nazwie KW8 (kod ten opisany jest w Dodatku C). Dane wejściowe na nośniku papierowym mogą być zapisane w dowolnym kodzie, który można przekodować na kod KW8. Uporządkowanie znaków dla danych jest takie jak kolejność znaków kodu KW8.

3.3.3.1.2. Struktura danych

Na każdym nośniku papierowym, to znaczy

- taśmie papierowej,
- kartach perforowanych,
- tabulogramie drukarki,

dane zapisane są w postaci strumienia znaków ze zbioru znaków dla danych.

Jednostki danych w strumieniu są następujące:

- pozycja - jest ona skończonym ciągiem znaków,
- dokument - jest on skończonym ciągiem pozycji,
- zbiór - jest on ciągiem dokumentów.

Te jednostki danych opisuje się w części DATA. Opis dokumentu składa się z opisów wszystkich jego pozycji.

Opis zbioru zawiera po jednym opisie dokumentu dla każdego typu dokumentu niezależnie od liczby dokumentów w zbiorze danych.

W strumieniu znaków mogą wystąpić następujące jednostki organizacyjne:

- ograniczniki danych,
- znaki pomocnicze,
- dokumenty specjalne.

Ograniczników danych używa się do oznaczania końca odpowiednich jednostek danych, aby umożliwić ich identyfikację. Ograniczniki danych są następujące:

- ogranicznik pozycji,
- ogranicznik dokumentu,
- ogranicznik zbioru.

Znaki pomocnicze wprowadzono w celu ułatwienia przygotowywania danych. Są one następujące:

- znak przeniesienia,
- znak wymazania.

Dokumenty specjalne są następujące:

- metryka początku zbioru,
- oznacznik operacji ponawiania,
- zerowator licznika dokumentów.

Metryka początku zbioru służy do identyfikacji zbioru. Pozostałe dokumenty specjalne związane są z operacją ponawiania programu (RERUN, RESTART).

W zależności od postaci danych na nośniku zbiory dzieli się na

- zbiory pozycyjne,
- zbiory niepozycyjne.

W zbiorach pozycyjnych każda pozycja musi zawierać dokładnie tyle znaków ile przewidziano w opisie tej pozycji w części DATA. Zbiory te nie posiadają ogranicznika pozycji.

W zbiorach niepozycyjnych pozycje nie muszą mieć ściśle określonej liczby znaków, ponieważ zakończone są ogranicznikiem pozycji.

W zależności od kierunku przesyłania danych zbiory dzieli się na

- zbiory wejściowe,
- zbiory wyjściowe.

Zbiory wejściowe mogą być pozycyjne lub niepozycyjne.

Zbiory wyjściowe są zawsze pozycyjne.

Tak zorganizowane dane wczytuje się do pamięci (instrukcja READ) lub w takiej postaci z niej wypisuje (instrukcja WRITE).

Jednorazowo wczytuje się lub wypisuje cały dokument. Przy wczytywaniu rozpoznawany jest jego typ, a pozycje rozmieszczane są w polu dokumentu zgodnie z ich opisem w części DATA.

Ograniczniki pozycji (o ile istnieją) oraz ograniczniki dokumentu nie są umieszczane w polu dokumentu. Ogranicznik zbioru traktowany jest jak dokument i jest wczytywany do pola dokumentu.

Przy wypisywaniu danych za każdą jednostką danych wypisywany jest odpowiedni ogranicznik.

Ogranicznikom danych nie odpowiadają żadne pozycje w polu dokumentu.

Ograniczniki danych ani dokumenty specjalne nie są opisywane w części DATA, mimo że ogranicznik zbioru oraz dokumenty specjalne są umieszczane w polu dokumentu.

Organizację danych oraz jednostki organizacyjne określa się zwrotem APPLY w części ENVIRONMENT.

3.3.3.1.3. Ograniczniki danych na nośnikach papierowych

Istnieją następujące rodzaje ograniczników danych:

- ogranicznik pozycji,
- ogranicznik dokumentu,
- ogranicznik zbioru.

Ogranicznik pozycji służy do oznaczania końca pozycji zbiorów niepozycyjnych. Ostatnia pozycja w dokumencie zbioru niepozycyjnego może nie być zakończona ogranicznikiem pozycji. Funkcję tę spełnia ogranicznik dokumentu. Ogranicznik dokumentu oznacza koniec dokumentu. Ostatni dokument zbioru może nie mieć ogranicznika dokumentu i może być bezpośrednio zakończony ogranicznikiem zbioru. Ogranicznik zbioru służy do oznaczania końca zbioru.

Ograniczniki danych mogą być standardowe lub zdefiniowane przez programistę.

Ogranicznik pozycji oraz dokumentu stanowi jeden znak. Ogranicznikiem zbioru jest ciąg następujących bezpośrednio po sobie czterech zna-

ków końca zbioru zakończony ogranicznikiem dokumentu.

Standardowe ograniczniki danych są następujące:

- ogranicznik pozycji - spacja
- ogranicznik dokumentu - nowa linia
- znak końca zbioru - znak równości

Niestandardowe ograniczniki danych można określić w zwrocie APPLY w części ENVIRONMENT.

Ciąg następujących po sobie jednakowych ograniczników danych jest traktowany jak jeden ogranicznik.

Gdy niestandardowy ogranicznik zostanie zdefiniowany, wówczas standardowy ogranicznik przestaje pełnić tę rolę i jest traktowany jak znak pozycji. Oczywiście żaden aktualny ogranicznik danych nie może wystąpić wśród znaków należących do pozycji.

Znaki pomocnicze

Znaki pomocnicze wprowadzono w celu ułatwienia przygotowania danych. Są one następujące:

- znak przeniesienia,
- znak wymazania.

Znak przeniesienia umożliwia przechodzenie do nowej linii. Anuluje on w danych wejściowych siebie i następujący bezpośrednio po nim znak nowej linii. Standardowym znakiem przeniesienia jest znak minus "-".

Znak wymazania służy do kasowania błędnie wyperforowanych danych. Wystąpienie tego znaku powoduje anulowanie:

- a) wszystkich znaków wczytanych po ostatnim ograniczniku pozycji - dla zbiorów niepozycyjnych,
- b) wszystkich znaków poprzedzających znak wymazania aktualnie wczytanego dokumentu - dla zbiorów pozycyjnych.

Standardowym znakiem wymazania jest znak odpowiadający w używanym kodzie zewnętrznym znakowi `' .DE/ '`.

Niestandardowe znaki pomocnicze można określić zwrotem `APPLY` w części `ENVIRONMENT`.

3.3.3.1.4. Dokumenty specjalne

3.3.3.1.4.1. M e t r y k a p o c z ą t k u z b i o r u

Zbiory danych mogą, ale nie muszą, posiadać metryki początku zbioru. Zależy to od tego czy w części `DATA` w zwrocie `LABEL` użyto opcji `LABEL OMITTED` czy `STANDARD`. Użycie opcji `LABEL OMITTED` oznacza, że zbiór metryki nie posiada, a użycie wersji `LABEL STANDARD`, że zbiór ma metrykę początku zbioru. Metryka początku zbioru wyjściowego tworzona jest w ramach wykonania instrukcji `OPEN` w części `PROCEDURE`. Natomiast metryka zbioru wejściowego jest wczytywana i sprawdzana w trakcie wykonania instrukcji `OPEN`.

Metryka początku zbioru musi być pierwszym dokumentem zbioru. Ma ona standardową postać i składa się z następujących dwu pozycji:

- identyfikator zbioru; jest to tekst zawierający 18 znaków,
- data utworzenia zbioru; jest ona ciągiem 5 cyfr; pierwsze dwie oznaczają rok, trzy pozostałe oznaczają dzień w roku.

Pozycje te są od siebie oddzielone ciągiem spacji. Cała metryka zakończona jest znakiem nowej linii.

Pozycjom metryki odpowiadają w programie zmienne IDENT i CREAT zwane pozycjami metrykowymi.

W pamięci mają one następującą budowę:

- identyfikator zbioru (IDENT) jest ciągiem 18 znaków spakowanych po trzy znaki w słowie;
- pozycja CREAT składa się z dwu liczb binarnych, umieszczonych w tym samym słowie tj.
 - rok w bitach 0-15,
 - dzień w roku w bitach 16-23;
- wartością zmiennej CREAT jest bieżąca data pobierana z systemu operacyjnego maszyny;
- wartość zmiennej IDENT można określić w zdaniu VALUE OF w części DATA DIVISION.

Ponadto w niestandardowej procedurze metrykowej wykonywanej w ramach zdania USE można nadawać wartości obydwu tym zmiennym lub wykorzystywać je na równi z innymi zmiennymi. O ile zmiennej IDENT nie zostanie nadana wartość, to przyjmuje się nazwę zbioru uzupełnioną spacjami lub obcięta do 18 znaków.

3.3.3.1.4.2. O z n a c z n i k o p e r a - c j i p o n a w i a n i a p r o g r a m u

Gdy w czasie czytania danych zostanie napotkany specjalny dokument zwany oznaczniakiem operacji

ponawiania, wówczas zostanie wykonana operacja RERUN, to znaczy stan całej maszyny (a więc zawartości rejestrów, pamięci operacyjnej oraz bębnowej i stan urządzeń zewnętrznych) zapamiętywany jest na wyznaczonej taśmie magnetycznej.

Oznacznik operacji ponawiania programu może występować tylko w zbiorach wejściowych, do których w zwrocie APPLY odnosi się opcja definiująca znak rerunowy (RERUN MARK). Oznacznik ten składa się z czterech znaków rerunowych określonych tą opcją, za którymi następuje ogranicznik dokumentu.

3.3.3.1.4.3. Z e r o w a t o r l i c z n i - k a d o k u m e n t ó w

W trakcie wykonywania operacji czytania liczne są wczytane dokumenty każdego zbioru. Każdy zbiór posiada swój licznik dokumentów.

Gdy w czasie czytania danych zostanie napotkany zerowator wówczas jego numer zostaje zapamiętany, a licznik dokumentów odpowiedniego zbioru zostaje wyzerowany. Wykorzystywane jest to przy operacji ponawiania programu RESTART.

Zerowator licznika dokumentów ma postać ciągu następujących bezpośrednio po sobie znaków:

- trzy znaki końca zbioru,
- liczba całkowita bez znaku,
- ogranicznik dokumentu.

Liczba całkowita jest numerem zerowatora.

3.3.3.1.5. Zbiory pozycyjne

Pozycyjność zbioru polega na tym, że miejsce znaku danych na nośniku określa jednoznacz-

nie miejsce tego znaku w polu dokumentu.

Zbiorami pozycyjnymi mogą być zbiory wejściowe oraz wyjściowe.

Zbiory pozycyjne nie posiadają bufora zbioru.

Znaki stanowiące dokument zbioru pozycyjnego wczytywane są bezpośrednio do pola dokumentu i umieszczane w nim kolejno jeden znak w jednym słowie na jego ostatnich ośmiu bitach.

Po drodze znaki przekodowywane są na kod wewnętrzny (żadne inne przekształcenia nie mają miejsca). Ogranicznik dokumentu nie jest umieszczany w polu dokumentu. Przy wypisywaniu odbywa się proces odwrotny: kolejne znaki z pola dokumentu umieszczane są na nośniku (w kodzie zewnętrznym), a po ostatnim znaku dokumentu dopisywany jest ogranicznik dokumentu. Zbiory pozycyjne nie posiadają ogranicznika pozycji. Każda pozycja musi być zapisana na nośniku w ten sposób, aby po umieszczeniu jej w polu dokumentu zajęła dokładnie tyle miejsca (słów) ile podaje w programie (w części DATA) opis tej pozycji.

W związku z tym pozycje zbioru pozycyjnego muszą być opisane zwrotem PICTURE, to znaczy zbiory pozycyjne muszą być znakowe. Pozycje takich zbiorów nie należy opisywać zwrotami FIXED, FIXED2, FLOAT.

Typ dokumentu zbioru pozycyjnego określony jest jego długością, to znaczy sumaryczną liczbą znaków pozycji tego dokumentu. Liczba znaków wczytanego dokumentu podawana jest do rejestru TALLY w trakcie operacji czytania (READ). Jeśli do zbioru należy kilka typów dokumentów, to muszą

być one różnej długości, bowiem jest to jedyny sposób rozpoznawania typu wczytywanego dokumentu zbioru pozycyjnego.

Pozycje liczbowe w zbiorach pozycyjnych wejściowych muszą być opisane obrazem liczbowym, (natomiast nie mogą być opisane obrazem liczbowym wydawniczym). W związku z tym nie mogą zawierać żadnych znaków poza cyframi oraz poprzedzającym je znakiem plus lub minus. Liczba musi być zapisana na nośniku w ten sposób, aby po wczytaniu do pola dokumentu miała skalę zgodną ze swoim obrazem. Pozycje liczbowe mogą zawierać co najwyżej 18 cyfr. Jeśli w obrazie wystąpi znak S, to na nośniku liczba musi być opatrzona znakiem plus lub minus. O ile znak S nie wystąpi, to na nośniku liczba nie może być poprzedzona znakiem.

Pozycja tekstowa może składać się z dowolnych znaków, które dają się przekodować na kod wewnętrzny (KW8). Pozycja musi być zapisana na nośniku wejściowym tak, aby po wczytaniu zajęła w polu dokumentu tyle słów, ile określa obraz tej pozycji. Na nośniku pozycji tekstowych nie obejmuje się apostrofami.

W zbiorach pozycyjnych wyjściowych pozycje dokumentu mogą być opisane obrazami wydawniczymi. Liczby na wyjściu mogą zawierać różne symbole wydawnicze.

3.3.3.1.6. Zbiory niepozycyjne

Zbiór niepozycyjny jest to taki zbiór, którego pozycje na nośniku nie muszą zawierać dokładnie tylu znaków, ile określa ich opis w części DATA.

W związku z tym każda pozycja takiego zbioru musi być zakończona na nośniku ogranicznikiem pozycji (po ostatniej pozycji może wystąpić od razu ogranicznik dokumentu).

Zbiorami niepozycyjnymi mogą być tylko zbiory wejściowe.

Pozycje zbiorów niepozycyjnych mogą być w części DATA opisane zwrotami FIXED, FIXED2, FLOAT, PICTURE. Zbiory niepozycyjne posiadają bufor zbioru.

Dane tworzące dokument (lub jego część) wczytywane są do bufora zbioru. Każdy znak wczytanej pozycji zajmuje 8 najmniej znaczących bitów kolejnego słowa buforu, ponadto słowo zawierające pierwszy znak pozycji ma jedynkę w zerowym bicie. W zależności od techniki czytania i długości buforu w buforze znajduje się cały dokument lub jego część; nie zajęte słowa buforu zostają wyzerowane. W buforze rozpoznawany jest typ wczytanego dokumentu, po czym poszczególne jego pozycje przekształcane są na postać wewnętrzną - zgodną z ich opisem w części DATA - i posyłane do pola dokumentu. Przekształcanie i przesyłanie pozycji odbywa się zgodnie z regułami przesyłania pozycji znakowych do obliczeniowych oraz do znakowych. W trakcie czytania usuwane są ograniczniki pozycji oraz dokumentu.

Należy zwrócić uwagę na to, że jeśli zwrot określający długość bufora nie wystąpi, to długość ta będzie równa długości pola dokumentu. Gdy pozycja opisana została w części DATA jako obliczeniowa (FIXED, FIXED2, FLOAT), to zarezerwowano dla niej jedno lub dwa słowa. Tymczasem w buforze pozycje umieszczane są znakowo (jeden znak zajmuje jedno słowo). W związku z

tym należy zarezerwować taki bufor, aby mieścił się w nim cały dokument lub taka jego część, która umożliwia rozpoznanie typu dokumentu (to znaczy zawierająca znacznik dokumentu).

W celu ułatwienia kontroli danych wejściowych dla każdego zbioru niepozycyjnego wprowadzono zmienną zastrzeżoną BUFF. Gdy w programie występuje więcej niż jeden zbiór niepozycyjny, wówczas przy odwołaniach do zmiennej BUFF należy kwalifikować ją nazwą zbioru. Zmienna BUFF jest znakowa o długości równej długości bufora zbioru. Adres tej zmiennej równy jest adresowi bufora zbioru, a wartość - aktualnemu stanowi bufora.

Pozycje liczbowe obok cyfr mogą zawierać kropkę dziesiętną oraz być poprzedzone znakiem plus lub minus niezależnie od wystąpienia znaku S w obrazie (o ile oczywiście są opisane obrazem). Nie nakłada się ograniczenia na długość pozycji na nośniku. Ale gdy w wyniku przekształcenia na postać wewnętrzną otrzyma się liczbę za długą w stosunku do zarezerwowanego dla niej miejsca w polu dokumentu, to wartość liczby może ulec zmianie.

Pozycje tekstowe mogą się składać z dowolnych znaków, które dają się przekodować na kod wewnętrzny (KWB).

Pozycji tekstowych nie obejmuje się apostrofami.

Jeżeli pozycja zapisana na nośniku jest za długa w stosunku do zarezerwowanego miejsca w polu dokumentu, to zostaje obcięta z prawej strony.

Taka organizacja danych (niepozycyjność) znakomicie ułatwia przygotowywanie danych na nośniku.

3.3.3.1.6.1. R o z p o z n a w a n i e t y - p u d o k u m e n t u z b i o - r u n i e p o z y c y j n e g o

Dokument jest typu n, jeżeli jego opis w części DATA jest n-ty z kolei. Po wczytaniu dokumentu i rozpoznaniu jego typu do rejestru TALLY posyłana jest liczba całkowita binarna określająca jego typ.

Ze względu na sposób rozpoznawania typu - dokumenty dzielimy na:

- dokumenty nie posiadające znacznika dokumentu; ich typ rozpoznaje się według ilości pozycji dokumentu;
- dokumenty posiadające znacznik dokumentu; ich typ rozpoznaje się według wartości znacznika dokumentu.

Znacznik dokumentu jest to taka pozycja dokumentu, której nadano wartość za pomocą zwrotu VALUE. Jeżeli zwrotem VALUE nadano wartość kilku pozycjom to znacznikiem jest pierwsza z nich.

Istnieją trzy sposoby rozpoznawania typu dokumentu. Programista może wybrać jeden z nich używając zwrotu APPLY z następującymi opcjami:

- A. UNPOSITIONAL 1 TECHNIQUE
- B. UNPOSITIONAL 2 TECHNIQUE
- C. UNPOSITIONAL 3 TECHNIQUE.

A. Sposób określany jako UNPOSITIONAL 1 TECHNIQUE polega na rozpoznawaniu typu za pomocą liczby pozycji w dokumencie. W takim zbiorze

każdy typ dokumentu musi posiadać inną ilość pozycji. Bufor takiego zbioru musi być tak określony (zwrotem BLOCK w części DATA) aby mieścić się w nim w całości najdłuższy dokument zbioru.

Jeżeli pozycja jest macierzą, to przy obliczaniu długości dokumentu uwzględniana jest każda jej składowa.

- B. Sposób określony jako UNPOSITIONAL 2 TECHNIQUE polega na rozpoznawaniu typu dokumentu za pomocą znacznika dokumentu. Bada się czy wartość pozycji opisanej jako znacznik jest zgodna z wartością zadeklarowaną zwrotem VALUE. W takim zbiorze dokumenty wszystkich typów muszą posiadać znaczniki. W buforze takiego zbioru nie musi mieścić się cały dokument. Wystarczy, aby mieściła się w nim pierwsza część dokumentu zawierająca znacznik. Wartości znaczników poszczególnych typów dokumentów muszą być różne.
- C. Sposób określony jako UNPOSITIONAL 3 TECHNIQUE stosuje się do zbiorów, w których część dokumentów opisana jest ze znacznikiem, a część bez znacznika. Wówczas pierwszy dokument opisany w zbiorze musi zawierać znacznik. Rozpoznawanie typu dokumentu odbywa się następująco: najpierw bada się czy wczytany dokument posiada znacznik odpowiadający znacznikowi, któregoś dokumentu w opisie, to znaczy stosuje się sposób B; jeżeli tak, to typ jest już określony; jeśli dokument nie posiada znacznika, to typ jest określany sposobem A to znaczy według długości w pozycji, ale bierze się pod uwagę tylko

menty nie posiadające znaczników, których opisy wystąpiły w części DATA między opisem dokumentu ostatnio rozpoznanego metodą B, a opisem następnego dokumentu zawierającego znacznik.

Stosując tę metodę rozpoznawania typów dokumenty na nośniku należy tak grupować, aby grupa dokumentów opisanych jako nieposiadające znacznika była bezpośrednio poprzedzona na nośniku dokumentem zawierającym znacznik, którego opis bezpośrednio poprzedza opisy dokumentów tej grupy w części DATA. Kolejność takich grup jest dowolna.

Bufor takiego zbioru musi być tak duży, aby mieścił się w nim:

- cały dokument, o ile nie zawiera znacznika,
- początkowa część dokumentu zawierająca znacznik, o ile dokument posiada znacznik.

Sposób C nosi nazwę techniki mieszanej.

Przykład

W części DATA opisano zbiór składający się z następujących dokumentów:

01 A.

02 B; PIC '99' VALUE 12

⋮

01 X.

} n pozycji elementarnych

⋮

01 Y.

} p pozycji elementarnych

⋮

01 Q. } q pozycji elementarnych
 ::}

01 K.

02 D; PIC '9' VALUE 3

::

01 M.

::} p pozycji elementarnych

01 N.

::} x pozycji elementarnych

01 W.

02 PIC '99' VALUE 17

::}

01 Z.

::} q pozycji elementarnych

01 E.

::} n pozycji elementarnych

Wczytywane są dokumenty w następującej kolejności:

- | | |
|----------------------------|------------|
| • ze znacznikiem równym 3 | TALLY = 5 |
| • bez znacznika; x pozycji | TALLY = 7 |
| • bez znacznika; p pozycji | TALLY = 6 |
| • bez znacznika; x pozycji | TALLY = 7 |
| • ze znacznikiem równym 12 | TALLY = 1 |
| • bez znacznika; p pozycji | TALLY = 3 |
| • bez znacznika; n pozycji | TALLY = 2 |
| • ze znacznikiem równym 17 | TALLY = 8 |
| • bez znacznika; n pozycji | TALLY = 10 |

3.3.3.2. Postać danych na taśmach magnetycz- nym

3.3.3.2.1. Postać i organizacja zbiorów taś- mowych

Jednostki danych przechowywanych na taśmach magnetycznych mają postać analogiczną jak postać jednostek danych w pamięci.

Informacje przechowywane są na taśmie magnetycznej blokami, które mogą się składać z co najmniej 5, a co najwyżej 2048 słów maszyny.

Pierwsze pięć słów każdego bloku stanowi metrykę bloku. Ma ona standardową budowę:

słowo 0 zawiera	-	kolejny numer bloku na szpulii taśmy, licząc od zera,
1	-	liczbę słów bloku, nie licząc metryki bloku,
2	-	sumę arytmetyczną słowa 0 i 1,
3 i 4	-	sumę kontrolną informacji tworzącej blok (nie licząc informacji tworzącej metrykę bloku).

Istnieją dwa rodzaje bloków:

- bloki danych,
- bloki organizacyjne.

Rodzaj bloku poznaje się po wartości zerowego słowa bloku. Słowo to jest ujemne dla bloków organizacyjnych i dodatnie dla bloków danych.

Blokiem danych jest zawartość bufora zbioru zwiększona o 5 słów metryki bloku. Długość bu-

fora, a tym samym bloku danych, opisywana jest zwrotem BLOCK w części DATA. O ile zwrot ten nie wystąpi - długość bufora równa jest długości pola dokumentu. Przy czytaniu, do rejestru TALLY posyłana jest liczba słów wczytanego dokumentu.

Bloki organizacyjne są następujące:

Metryka początku zbioru	(MPZb)
Metryka końca zbioru	(MKZb)
Metryka początku szpuli	(MPSz)
Metryka końca szpuli	(MKSz)
Metryka końca zapisu	(MKZa)
Blok RERUN	(BR)

Kolejność występowania poszczególnych rodzajów bloków zależy od tego w jakim stosunku pozostaje do siebie zbiór danych i szpula taśmy. Istnieją trzy rodzaje zbiorów:

- jednoszpulowy, tzn. z jedną szpulą związany jest jeden zbiór; kolejność bloków na taśmie jest taka:

MPS	Metryka początku szpuli
MPZb	Metryka początku zbioru
BDR	Blok danych lub blok RERUN
:	
MKZb	Metryka końca zbioru
MKZa	Metryka końca zapisu

- wielozbiorowy, tzn. na jednej szpuli taśmy znajduje się kilka zbiorów; zbiory takie muszą być opisane zwrotem MULTIPLE w części ENVIRONMENT; daty ich ważności (EXPIR) muszą tworzyć ciąg nierosnący (najszybciej powinna upłynąć ważność ostatnio zapisanego zbioru),

umożliwia to dopisywanie nowych zbiorów na koniec taśmy. Kolejność bloków na taśmie jest taka:

I zbiór	{	MPS
		MPZb
		BDR
		⋮
		MKZb
II zbiór	{	MPZb
		BDR
		⋮
		MKZb
		⋮
		MKZa

• wieloszpulowy, tzn. jeden zbiór zajmuje kilka szpul taśmy; kolejność bloków:

I szpula	{	MPS
		MPZb
		BDR
		⋮
		MKS
II szpula	{	MPS
		MPZb
		BDR
		⋮
		MKS
		⋮

ostatnia szpula	{	MPS
		MPZb
		BDR
		⋮
		MKZb
		MKZa

Aby zmniejszyć prawdopodobieństwo powstania błędu zapisu na taśmie magnetycznej, zaleca się kilkakrotne zapisanie tego samego bloku na taśmie. Liczbę powtórzeń zapisu każdego bloku deklaruje się w zwrocie APPLY w części ENVIRONMENT. Jeżeli jeden zapis nie daje się odczytać - próbuje się odczytać następny zapis tego bloku. Po wyczerpaniu się zapisów bloku sygnalizowany jest błąd.

Zbiory na taśmach magnetycznych mogą być wejściowe oraz wyjściowe.

Metryki początku i końca zbiorów wyjściowych są tworzone przez instrukcje OPEN oraz CLOSE, a metryki zbiorów wejściowych są wczytywane oraz sprawdzane przez te instrukcje.

Jeśli jednak zbiorowi związanemu z taśmą magnetyczną przypisano w zwrocie LABEL własność OMITTED i jest on zbiorem wejściowym, to nie jest sprawdzana poprawność żadnej pozycji metryki początku zbioru, mimo że metryka ta jest wczytywana. Zbiory wyjściowe otrzymują metryki niezależnie od wystąpienia własności OMITTED. Dla zbiorów posiadających własność OMITTED nie są również wykonywane niestandardowe procedury metrykowe.

3.3.3.2.2. Metryka początku zbioru

Metryka początku zbioru jest blokiem zapisywanym na szpuli w wyniku wykonania instrukcji OPEN OUTPUT i sprawdzanym w trakcie wykonywania instrukcji OPEN INPUT. Metryka zawiera następujące pozycje dostępne w programie:

- identyfikator zbioru (pozycja metryki o nazwie IDENT),
- data zapisania zbioru na taśmie (pozycja CREAT),
- data ważności zbioru (pozycja EXPIR),
- numer generacji zbioru (pozycja GENER),
- numer sekcji zbioru (pozycja SECT).

Inne pozycje metryki nie są dostępne dla programisty.

Poszczególne pozycje metryki początku zbioru mają następującą budowę oraz znaczenie:

- identyfikator zbioru (IDENT) jest tekstem zawierającym 18 znaków; zapisywany jest on po trzy znaki w słowie; służy do identyfikacji zbioru;
- data utworzenia zbioru (CREAT) oraz data ważności zbioru (EXPIR) składa się z dwu części: liczby oznaczającej rok oraz liczby oznaczającej dzień w roku. Obie te liczby (rok i dzień) umieszczane są w jednym słowie w postaci binarnej:

- rok w bitach 0 - 15,
- dzień w bitach 16 - 23;

data ważności zbioru oznacza pierwszy dzień, w którym zbiór może być wymazany z taśmy;

- numer generacji zbioru (GENER) jest liczbą naturalną ≥ 1 , która określa kolejną wersję tego samego zbioru; zapisany jest on binarnie w jednym słowie;
- numer sekcji (SECT) jest liczbą naturalną ≥ 1 i ma istotne znaczenie dla zbiorów związanych z kilkoma szpulami taśmy. Sekcję stanowi część zbioru mieszcząca się na jednej szpuli. Na początku każdej z tych szpul zapisywana jest metryka początku zbioru, przy czym numer sekcji jest w każdej następnej szpuli większy o 1 od numeru sekcji na szpuli poprzedniej. Zbiory jednoszpułowe tworzą jedną sekcję. Numer sekcji umieszczany jest w jednym słowie w postaci binarnej.

Poza wymienionymi pozycjami z metryką początku zbioru związana jest jeszcze pewna zmienna nie będąca pozycją w metryce, ale zaliczana w programie do pozycji metrykowych. Jest to zmienna SECUR. Oznacza ona liczbę dni ważności zbioru i służy, wraz z pozycją CREAT, do obliczania daty ważności zbioru.

Wartości pozycji metrykowych można określić w programie w następujący sposób:

- pozycjom metrykowym IDENT, SECT, SECUR nadawać można wartości za pomocą zwrotu VALUE OF (DATA DIVISION);
- wartość pozycji GENER można wprowadzić z monitora w chwili otwarcia zbioru; wartość tę wprowadza się jako ciąg 5 cyfr zakończonych gwiazdką; wówczas w części DATA musi wystąpić zwrot LABEL STANDARD WITH GENER;
- wartością pozycji CREAT jest bieżąca data podawana przez system operacyjny maszyny;

- wartość pozycji EXPIR jest wyznaczana z wartości pozycji CREAT i SECUR;
- ponadto zmiennym IDENT, SECT, CREAT, EXPIR, GENER można nadawać wartości za pomocą niestandardowych procedur metrykowych, określonych w zdaniu USE. Wówczas pozycje te traktowane są jak pozycje elementarne o następujących własnościach:

IDENT	PICTURE	'X(18)'
CREAT	FIXED	
GENER	FIXED	
EXPIR	FIXED	
SECT	FIXED	

Jeśli pozycjom metrykowym nie zostaną nadane wartości przez programistę, to przyjmuje się następujące wartości domyślne:

- IDENT - nazwa zbioru uzupełniona spacjami lub obcięta do 18 znaków,
- SECT - całkowita 1,
- SECUR - całkowita 0,
- GENER - całkowita 1.

3.3.3.2.3. Metryka końca zbioru

Metryka końca zbioru jest ostatnim blokiem zbioru. Zawiera ona informacje umożliwiające automatyczne sprawdzenie poprawności odczytu danych zbioru. Metryka końca zbioru powstaje w wyniku wykonania instrukcji CLOSE.

3.3.3.2.4. Metryka początku szpuli

Metryka początku szpuli jest pierwszym blokiem na każdej szpuli taśmy. Tworzy się ją i spraw-

dza poza językiem COBOL. Za pomocą tej metryki można identyfikować szpule oraz wprowadzać ograniczenia w dostępie do nich.

3.3.3.2.5. Metryka końca szpuli

Metryka końca szpuli jest ostatnim blokiem zapisywanym na szpuli. Sygnalizuje ona, że na tej szpuli nie można zapisać żadnej nowej informacji, dopóki nie upłynie termin ważności ostatniego zbioru zapisanego na niej. Metryka końca szpuli tworzona jest automatycznie przy przejściu od szpuli do szpuli oraz w wyniku operacji CLOSE REEL.

3.3.3.2.6. Metryka końca zapisu

Metryka końca zapisu tworzona jest w wyniku wykonania instrukcji CLOSE i znajduje się za ostatnim zapisanym blokiem. Jeśli metryki tej nie ma na taśmie, to nie można zapisywać na niej nowych zbiorów. Na taśmach nie zapisanych powinna się ona znaleźć bezpośrednio za metryką początku szpuli. Poza językiem COBOL metrykę tę umieścić można za pomocą tak zwanej operacji metrykowania.

3.3.3.2.7. Blok RERUN

Blok RERUN wykorzystywany jest przy operacji ponawiania programu. Powstaje on w wyniku operacji RERUN. Grupa bloków RERUN zawiera wszystkie informacje niezbędne do wznawiania wyko-

nywania programu od stanu jaki był w momencie dokonania ostatniej operacji RERUN przed przerwaniem pracy programu. Wykonywanie operacji RERUN określane jest zwrotem RERUN.

4. B U D O W A P R O G R A M U

4.1. IDENTIFICATION DIVISION - IDENTYFIKACJA PROGRAMU

Postać ogólna części identyfikującej - zob. Wkładka I, pkt 1.

Część ta służy do identyfikacji programu.

<nazwa-programu> jest ciągiem dowolnych znaków ze zbioru znaków dla języka - nie zawierającym kropki. (Nie może więc być identyfikatorem ciąg zawierający łącznik '..') Kropka występująca za nazwą wyznacza koniec nazwy. Nazwa programu wypisywana jest na urządzenie określone jako zerowe wyjście translatora w chwili, gdy zaczyna się translacja tego programu. Cztery pierwsze znaki nazwy są nazwą programu binarnego, gdy program taki będzie produkowany. Nie może wtedy wśród nich wystąpić litera O. <tekst-komentarzowy> jest ciągiem dowolnych znaków ze zbioru znaków języka nie zawierającym następujących słów zastrzeżonych: ENVIRONMENT, DATA, PROCEDURE, END.

Tekst ten jest całkowicie pomijany przez translator.

Przykład

IDENTIFICATION DIVISION.

PROGRAM..ID.OBLICZANIE FORMUL.

WERSJA WSTEPNA. PIERWSZA CZESC.

ENVIRONMENT DIVISION.

Po wczytaniu tej części programu wypisany zostanie tekst OBLICZANIE FORMUL.

Natomiast tekst: WERSJA WSTEPNA, CZESC PIERWSZA zostanie wczytany, ale zignorowany.

4.2. ENVIRONMENT DIVISION - URZADZENIA

Postać ogólna części ENVIRONMENT podana jest we wklejce.

Zwrot COPY <nazwa-biblioteczna> opisany jest w części PROCEDURE.

4.2.1. CONFIGURATION SECTION - SEKCJA KONFIGURACJI

W sekcji tej podaje się:

- nazwę maszyny źródłowej, tzn. maszyny, na której program ma być tłumaczony (paragraf SOURCE..COMPUTER),
- nazwę maszyny roboczej, tzn. nazwę maszyny, na której program ma być wykonywany (paragraf OBJECT..COMPUTER) oraz wielkość pamięci dla programu (zwrot MEMORY).

W opisywanej implementacji języka COBOL paragrafy te są ignorowane przez translator bowiem programista nie ma wyboru maszyny (jest nią ZAM 41).

Wielkość pamięci natomiast jest określana przez operatora (z monitora) w czasie sprowadzania systemu operacyjnego.

W przypadku używania taśmy binarnej program musi być wykonywany na takiej pamięci na jakiej taśma binarna była produkowana.

Ponadto w sekcji tej podaje się:

- niestandardowy znak waluty (zwrot CURRENCY),
- numery wskaźników wraz z nazwami przyporządkowanymi ich stanom (opis wskaźnika).

4.2.1.1. SWITCH - opis wskaźnika

Postać

SWITCH<całkowita> $\left[\begin{array}{l} \text{ON STATUS IS <nazwa-wskaźnika>} \\ \text{OFF STATUS IS <nazwa-wskaźnika>} \end{array} \right]$

gdzie:

<nazwa-wskaźnika> jest identyfikatorem.

<całkowita> jest numerem wskaźnika i może przyjmować wartości od 0 do 23. Wskaźnik o numerze n jest to n-ty bit trzeciego słowa maszyny. Każdy wskaźnik może przybierać wartości 0 lub 1.

Wartość wskaźnika można ustawiać za pomocą dyrektywy USTAW WSKAZNIK lub dyrektywy WPISZ DO PAO języka JOS.

<nazwa-wskaźnika> jest nazwą wartości jaką przybiera wskaźnik. Nazwa ta traktowana jest jako zmienna przybierająca wartości logiczne "prawda" lub "fałsz" w zależności od tego, którym zwrotem (ON lub OFF) została opisana oraz od wartości jaką przybiera wskaźnik.

Zależność wartości nazwy wskaźnika od wartości wskaźnika oraz zwrotu, którym nazwa została zadeklarowana jest następująca:

Wartość wskaźnika	Wartość nazwy zadeklarowanej zwrotem	
	ON STATUS	OFF STATUS
0	prawda	fałsz
1	fałsz	prawda

Nazwy wskaźnika można używać w części PROCEDURE wszędzie tam, gdzie może wystąpić warunek.

Przykład

SWITCH 5 ON STATUS A1; OFF B3D

A1 oraz B3D są tu nazwami wskaźnika o numerze 5. Jeżeli bit 5 słowa trzeciego ma wartość 1, to A1 ma wartość "fałsz", a B3D ma wartość "prawda". Gdy 5 bit słowa trzeciego ma wartość zero, to A1 ma wartość "prawda" a B3D ma wartość "fałsz".

4.2.1.2. Zwrot CURRENCY

Postać

CURRENCY SIGN IS 'znak-waluty'

<znak waluty> musi być pojedynczym znakiem kodu wewnętrznego (KW8).

Zwrot ten określa niestandardowy znak waluty. Znak ten wstawiany jest do pola pozycji, która została opisana obrazem liczbowym wydawniczym zawierającym znak W. W samym obrazie jednak może wystąpić tylko znak W.

O ile zwrot CURRENCY nie wystąpi, to znakiem waluty wstawianym do pola pozycji jest litera W.

Przykład

CURRENCY SIGN IS '*'

Zwrot ten oznacza, że niestandardowym znakiem waluty jest gwiazdka.

4.2.2. INPUT..OUTPUT SECTION-SEKCJA WEJŚCIA-WYJŚCIA

Sekcja ta składa się z dwu paragrafów.

W paragrafie FILE..CONTROL przydziela się urządzenia zewnętrzne zbiorom danych oraz zbiorom sortowania (deklaracja SELECT).

W paragrafie I..O..CONTROL podaje się informacje związane z organizacją przesyłania danych, a mianowicie:

- definiuje się sposób wczytywania lub wypisywania danych oraz związaną z tym organizację zbioru danych (zwrot APPLY),
- określa się, które zbiory mogą mieć wspólne pole pamięci (pole dokumentu lub bufor) (zwrot SAME),
- podaje się nazwy zbiorów, które związane są z tą samą szpulą taśmy magnetycznej (zwrot MULTIPLE),
- określa się parametry operacji RERUN związanej z ponawianiem programu (zwrot RERUN).

4.2.2.1. | Paragraf FILE..CONTROL
Deklaracja SELECT

Deklaracja ta ma trzy wersje związane z rodzajem zbioru.

Wersja 1. Dotyczy zbiorów danych

Postać podana jest we wkładce.

Wersja 2. Dotyczy zbiorów sortowania

Postać

SELECT <nazwa-zbioru-sortowania>

.ASSIGN TO [MTAPE <litera>]

Wersja 3. Dotyczy zbiorów danych będących
wynikiem operacji sortowania

Postać

SELECT <nazwa-zbioru-danych> ASSIGN TO

{ MTAPE <litera>
[MTAPE <litera>] OR MTAPE <litera> }

gdzie:

<nazwa-zbioru-danych> jest identyfikatorem,
<nazwa-zbioru-sortowania> jest identyfikatorem,
<litera> jest jedną z liter A,B,C,D,E,F.

Za pomocą deklaracji SELECT przydziela się urządzenia zewnętrzne zbiorom danych lub zbiorom sortowania.

Do każdego zbioru opisanego w części DATA musi odnosić się jedna i tylko jedna deklaracja SELECT. Z drugiej strony każdy zbiór, do którego

odnosi się deklaracja SELECT, musi być opisany w części DATA.

Wersja 1

Deklaracja SELECT w tej wersji służy do przydzielania urządzeń zewnętrznych zbiorom danych, a także dodatkowych buforów zbiorom związanym z taśmą magnetyczną.

Użycie zwrotu MTAPE <litera> oznacza, że zbiór jest związany z taśmą magnetyczną i że przydzielono mu stację lub kilka stacji taśmy magnetycznej, które w konkretnym zestawie maszyny mają symbol <litera>. Przyporządkowanie liter numerom stacji magnetycznych jest następujące:

A	oznacza	8, 9
B	"	10, 11
C	"	12, 13
D	"	14, 15
E	"	16, 17
F	"	18, 19

Zob. ZAM 41 OPROGRAMOWANIE (SYSTEM 141), t. 1, MASZYNA UŻYTKOWA.

Zbiorowi związanemu z taśmą magnetyczną wolno przydzielić co najwyżej cztery stacje taśmy magnetycznej. Zbiorowi takiemu można również przydzielić dodatkowe bufony. Służy do tego zwrot RESERVE, w którym <całkowita> oznacza liczbę dodatkowych buforów. Liczba ta powinna wynosić 1. Przydzielenie innej liczby dodatkowych buforów jest dozwolone, ale nie ma znaczenia.

Przydzielenie jednego dodatkowego buforu powoduje przyspieszenie operacji czytania z taśmy magnetycznej lub pisanie na niej. Użycie wersji RESERVE w wersji NO lub brak tego zwrotu - spowoduje nieprzydzielenie dodatkowego buforu. Wielkość bufora określana jest zwrotem BLOCK w części DATA.

Zbiór danych, któremu urządzenie zewnętrzne przydzielono zwrotem INPUT <całkowita> jest zbiorem wejściowym związanym z nośnikiem papierowym:

- taśmą papierową,
- kartami,

a przydzielone mu urządzenie może być:

- czytnikiem taśmy papierowej,
- czytnikiem kart.

Zbiór danych, któremu urządzenie zewnętrzne przydzielono zwrotem OUTPUT <całkowita> związany jest z nośnikiem papierowym, to znaczy:

- z taśmą papierową,
- z tabulogramem drukarki,

a odpowiednie urządzenie zewnętrzne, które mu przydzielono, może być:

- perforatorem taśmy papierowej,
- drukarką wierszową.

Odpowiednie urządzenie określone jest symbolicznym numerem wejścia - wyjścia, <całkowita> oznacza numer symbolicznego wejścia-wyjścia i może przyjmować wartości od 0 do 15.

Ten sam numer symboliczny można przydzielić dwóm urządzeniom tylko wtedy, gdy jedno z nich

jest wyjściowe, a drugie wejściowe. Numery symbolicznych wejść-wyjść muszą być zdefiniowane poza językiem COBOL w czołówce programu (Język JOM). Ponadto dla symbolicznych wejść-wyjść należy dołączyć:

- podprogram czytania lub pisania dla urządzenia zewnętrznego,
- odpowiedni dekodery

oraz zarezerwować niezbędne pole pamięci. Dokonuje się tego za pomocą wstawki w języku SAS w części proceduralnej programu. Dołączania ani odłączania symbolicznych wejść-wyjść nie wykonują żadne instrukcje COBOL-u. Dokładne reguły dotyczące dołączania wejść-wyjść podane są w ZAM 41, OPROGRAMOWANIE (SYSTEM 141), t. 1, MASZYNA UŻYTKOWA.

Wersja 2

Deklaracja SELECT w tej wersji służy do przydzielania stacji taśmy magnetycznej zbiorom sortowania. Na stacjach tych odbywa się operacja sortowania. Zbiorowi sortowania można przydzielić od 3 do 6 stacji taśmy magnetycznej.

O ile zbiór danych podlegający sortowaniu znajduje się na taśmie magnetycznej aktualnie zawieszony na stacji, która ma być jednocześnie użyta do sortowania, to przydzielając stację zbiorowi sortowania należy tę stację wymienić na liście deklaracji SELECT (wersja 2) jako ostatnią. Dotyczy to nazwy zbioru danych użytego w zwrocie USING operacji SORT.

Wersja 3

Deklaracja SELECT w wersji 3 służy do przydzielania stacji taśmy magnetycznej zbiorowi danych, którego nazwa występuje w zwrocie GIVING instrukcji SORT. Zbiór ten zawiera dokumenty w posortowanej kolejności.

W zależności od tego, czy posortowane dokumenty zbioru mają pozostać na jednej ze stacji przydzielonej zbiorowi sortowania, czy też mają być przesłane na inną stację taśmy magnetycznej, w deklaracji tej wymienia się odpowiednio:

- te same stacje (wszystkie), które przydzielono zbiorowi sortowania,
- jedną stację aktualnie nie zajętą przez żaden inny zbiór.

O ile zbiorowi danych przydzielili się te same stacje co zbiorowi sortowania, to nie można z góry określić, gdzie znajdą się posortowane dokumenty. Zależy to od przebiegu operacji sortowania. O wyniku sortowania informuje odpowiedni monit.

Urządzenie przydzielone zbiorowi za pomocą zwrotu SELECT można zmienić za pomocą zmiennej zastrzeżonej HARDWARE. Jest to zmienna typu FIXED, której wartością w czasie wykonywania programu jest numer urządzenia ostatnio przydzielonego zbiorowi. Zmienna ta może występować wszędzie tam, gdzie może występować argument typu FIXED.

Nazwom urządzeń odpowiadają następujące wartości zmiennej HARDWARE:

- dla symbolicznych we-wy - numer symbolicznego we-wy,
- dla zbiorów związanych z TM:

A-4

B-5

C-6

D-7

E-8

F-9

Jeżeli zbiorowi przydzielono więcej niż jedno urządzenie, to numery tych urządzeń rozmieszczone są w polu zmiennej **HARDWARE** następująco:

HARDWARE	!	q4	!	q3	!	q2	!	q1	!
----------	---	----	---	----	---	----	---	----	---

gdzie q_i są numerami kolejnych urządzeń.

Dla zbioru sortowania tworzy się n zbiorów pomocniczych, gdzie n jest liczbą zadeklarowanych urządzeń.

Numery tych urządzeń zawierają zmienne zastrzeżone o nazwach

HARDWARE1, **HARDWARE2**, ..., **HARDWAREn**.

Jeżeli w programie występuje więcej niż jeden zbiór, to nazwę **HARDWARE** lub **HARDWAREi** należy kwalifikować nazwą zbioru.

Zmiany numeru urządzeń dla zbioru można dokonać tylko przed kolejnym otwarciem zbioru.

Przykłady

Wersja 1

SELECT OBSZAR ASSIGN TO MTAPE B, MTAPE C.

Zbiór OBSZAR jest zbiorem danych związanych z taśmą magnetyczną; przydziela mu się stacje B oraz C.

SELECT ADX MTAPE A; RESERVE 1.

Zbiór ADX jest zbiorem danych związanym z taśmą magnetyczną; przydziela mu się stację A oraz dodatkowy bufor.

SELECT WEJ ASSIGN INPUT 4

Zbiór WEJ jest wejściowym zbiorem danych związanym z nośnikiem papierowym. Numerem symbolicznego wejścia jest 4.

Wersja 2

SELECT BCG MTAPE A, MTAPE B, MTAPE C, MTAPE D.

Zbiór BCG jest zbiorem sortowania. Sortowanie ma odbywać się na stacjach A, B, C i D.

Wersja 3

SELECT PYRY MTAPE A, MTAPE B, MTAPE C OR MTAPE D

Zbiór danych PYRY jest zbiorem otrzymanym na jednej ze stacji A, B, C lub D w wyniku operacji sortowania.

Zmienić urządzenie przydzielone zbiorowi ZBIOR..1 można następująco:

ACCEPT HARDWARE IN ZBIOR..1 FROM INPUT 1

Nowy numer urządzenia dla zbioru ZBIOR..1 zostanie wczytany z wejścia 1.

4.2.2.2. Paragraf I..0..CONTROL

4.2.2.2.1. Zwrot APPLY

Postać

APPLY <technika-wejścia-wyjścia>
ON [] <nazwa-zbioru-danych> []

<nazwa-zbioru-danych> jest identyfikatorem
<technika-wejścia-wyjścia> ma postać zależną
od rodzaju zbioru danych, którego dotyczy.

Wersja STANDARD dotyczy zbioru związanego
z taśmą magnetyczną

Postać

STANDARD [<całkowita>] TECHNIQUE

Wersja POSITIONAL dotyczy zbioru związanego
z nośnikiem papierowym - pozycyjnego

Postać - zob. wkładka po rozdz. 4.

Wersja UNPOSITIONAL dotyczy zbioru związanego
z nośnikiem papierowym - niepozycyjnego

Postać - zob. wkładka po rozdz. 4.

<znak> oznacza jeden znak kodu wewnętrznego
(KW8).

Zwrot APPLY definiuje pewne własności wymienio-
nych w tym zwrocie zbiorów danych. Własności te
są następujące:

- sposób przesyłania danych,
- postać danych na nośniku,
- sposób rozpoznawania typu dokumentu,
- niestandardowe ograniczniki danych oraz
znaki pomocnicze, znak rerunowy.

Wersja STANDARD oznacza, że zbiór związany jest z taśmą magnetyczną. Informacją przesyłaną jednorazowo z taśmy magnetycznej lub na taśmę jest blok będący zawartością bufora zbioru. Ze względu na prawdopodobieństwo powstawania błędów w zapisie lub odczycie bloku stosuje się kilkakrotny zapis tego samego bloku na taśmie magnetycznej. Wielkość <całkowita> określa ilość powtórzeń bloku na taśmie. Jeśli ilość ta nie zostanie określona w zwrocie APPLY, to stosuje się jednokrotny zapis bloku na taśmie. Wielkość bloku na taśmie magnetycznej określa się zwrotem BLOCK w części DATA.

Wersja POSITIONAL dotyczy zbiorów związanych z nośnikiem papierowym i oznacza, że zbiory danych, których nazwy wymieniono w tej wersji zwrotu APPLY, są pozycyjne. Pozycyjnymi mogą być zbiory zarówno wejściowe, jak i wyjściowe.

Zbiór pozycyjny nie posiada ogranicznika pozycji.

Wczytywanie techniką pozycyjną polega na tym, że znaki stanowiące dokument zbioru wczytywane są bezpośrednio do pola dokumentu i umieszczane w nim kolejno - jeden znak w jednym słowie pola dokumentu bez żadnych przekształceń. Ogranicznik dokumentu nie jest umieszczany w polu dokumentu. W rejestrze TALLY umieszczana jest liczba wczytanych znaków dokumentu. Przy wypisywaniu dokumentów zbiorów pozycyjnych odbywa się proces odwrotny. Kolejne znaki z pola dokumentu umieszczane są na nośniku. Za ostatnim znakiem dokumentu dopisywany jest

ogranicznik dokumentu. Należy pamiętać o tym, że o ile urządzeniem wyjściowym jest drukarka wierszowa, to jeśli długość wiersza przekroczy 120 znaków - przejście do nowego wiersza dokonywane jest automatycznie poza językiem COBOL.

Pozycje zbiorów pozycyjnych muszą być w części DATA opisane zwrotem PICTURE.

Wersja UNPOSITIONAL dotyczy zbiorów związanych z nośnikiem papierowym i oznacza, że zbiory danych wymienione w tym zwrocie są niepozycyjne. Zbiorami niepozycyjnymi mogą być tylko zbiory wejściowe. Niepozycyjność oznacza, że pozycje dokumentów zbioru zakończone są na nośniku ogranicznikiem pozycji. W związku z tym postać ich na nośniku może różnić się od postaci w polu dokumentu. Dane stanowiące dokument takiego zbioru (lub część dokumentu) wczytywane są do bufora zbioru. Tam rozpoznawany jest typ dokumentu i dopiero wtedy dane przekształcane są na postać zgodną z ich opisem w części DATA i przesyłane do pola dokumentu zgodnie z regułami przesyłania pozycji. Pozycje takiego zbioru mogą być zarówno znakowe (opisane zwrotem PICTURE), jak i obliczeniowe (opisane zwrotem FIXED, FIXED2, FLOAT). Przy wczytywaniu dokumentu liczba określająca typ dokumentu umieszczana jest w rejestrze TALLY.

Typem dokumentu zbioru niepozycyjnego jest liczba wskazująca kolejne miejsce opisu dokumentu w opisie danego zbioru w części DATA.

Sposób wczytywania dokumentu i rozpoznawania jego typu określa liczba całkowita występująca za słowem UNPOSITIONAL w zwrocie APPLY.

- 1 - oznacza, że typ dokumentu rozpoznawany jest tylko na podstawie ilości pozycji dokumentu
- 2 - oznacza, że typ dokumentu rozpoznawany jest na podstawie znacznika dokumentu. Znacznikiem dokumentu jest pierwsza z tych pozycji dokumentu, którym nadano wartość zwrotem VALUE w jej opisie w części DATA
- 3 - oznacza, że zbiór zawiera dokumenty posiadające znacznik oraz dokumenty bez znacznika. Wówczas na nośniku grupa dokumentów nie posiadających znacznika musi być poprzedzona dokumentem posiadającym znacznik, którego opis w części DATA bezpośrednio poprzedza opisy dokumentów z grupy dokumentów bez znacznika. Kolejność dokumentów w tej grupie jest już dowolna. Metoda 3 rozpoznawania typu polega na tym, że do bieżącego wczytywanego dokumentu stosuje się metodę B, a o ile nie da to rezultatu, wówczas stosuje się metodę A. Brane są tu jednak pod uwagę tylko opisy dokumentów, które wystąpiły w części DATA między opisem dokumentu ostatnio rozpoznanym na podstawie znacznika, a opisem następnego dokumentu posiadającego znacznik (porównaj pkt 3.3.3.1.6.1).

Niestandardowe ograniczniki danych, znaki pomocnicze oraz znacznik operacji ponawiania definiuje się następującymi opcjami:

FILE	- znak końca zbioru,
RECORD	- ogranicznik dokumentu,
ITEM	- ogranicznik pozycji (tylko dla zbiorów niepozycyjnych),
DELETION	- znak wymazania (tylko dla zbiorów niepozycyjnych),
CONTINUATION	- znak przeniesienia,
RERUN	- znak rerunowy.

Ogranicznik zbioru jest dokumentem złożonym z czterech znaków końca zbioru oraz ogranicznika dokumentu. Służy on do oznaczania końca zbioru.

Ograniczniki dokumentu oraz pozycji są pojedynczymi znakami. Służą do oznaczania odpowiednio końca dokumentu oraz końca pozycji.

Oznacznik operacji ponawiania jest dokumentem zbudowanym z czterech znaków rerunowych oraz ogranicznika dokumentu. Napotkanie takiego dokumentu w danych wejściowych powoduje zapisanie na taśmie magnetycznej stanu maszyny zgodnie z regułami dotyczącymi operacji ponawiania programu.

Znak wymazania służy do anulowania błędnie wyperforowanych danych. Wystąpienie jego wśród znaków danych powoduje anulowanie ostatnio wczytanej jednostki (jej początkowych znaków):

- w zbiorach pozycyjnych - całego dokumentu,
- w zbiorach niepozycyjnych - pozycji.

Znak przeniesienia umożliwia przejście do nowego wiersza, gdy znak nowej linii ma nie być brany pod uwagę. Znak przeniesienia znosi w danych wejściowych siebie i następujący bezpośrednio po nim znak nowej linii.

Jeżeli dla jakiegoś zbioru nie wystąpi zwrot APPLY lub nie wystąpią pewne jego opcje, to odpowiednie parametry określa się w zależności od urządzenia zewnętrznego, z którym związany jest zbiór.

Obrazuje to następująca tabelka:

Parametry	Urządzenia		
	INPUT	OUTPUT	MTAPE
technika we-wy	UNPOSITIONAL1	POSITIONAL	STANDARD1
ogranicznik pozycji	spacja	brak	brak
ogranicznik dokumentu	nowa linia	nowa linia	brak
znak końca zbioru	=	=	brak
znak przeniesienia	-	brak	brak
znak wymazania	' .DE' 1)	' .DE' 1)	brak
znak rerunowy	brak	brak	brak

O ile zdefiniowany zostanie niestandardowy znak, to odpowiedni standardowy znak przestaje pełnić swoją rolę i traktowany jest w danych jako znak pozycji.

Aby zlikwidować standardowy znak nie wprowadzając w to miejsce żadnego znaku niestandardowego należy jako znak niestandardowy zadeklarować znak odpowiadający znakowi ' .NU' lub ' .CR' w KW8, bowiem znaki te są ignorowane w trakcie czytania.

Przykłady

APPLY STANDARD 5 ON ALA, BELA, PELA

1) znak używanego kodu odpowiadający ' .DE' w KW8.

Zbiory ALA, BELA, PELA są związane z taśmą magnetyczną. Zapis bloku na taśmie magnetycznej jest pięciokrotny.

APPLY UNPOSITIONAL 2 FILE MARK '*', RERUN
MARK '+' ON K, R2, F

Zbiory K, R2, F są zbiorami związanymi z nośnikiem papierowym. Wszystkie dokumenty tych zbiorów mają znaczniki. Znakiem końca zbioru jest *, a znakiem rerunowym znak +. Pozostałe ograniczniki i znaki są standardowe.

4.2.2.2.2. Zwrot SAME

Postać

SAME [RECORD] AREA FOR ['] <nazwa-zbioru> []

gdzie <nazwa-zbioru> jest identyfikatorem.

Zwrot SAME określa, że zbiory wymienione na liście tego zwrotu mają mieć przydzielone wspólne obszary pamięci (pole dokumentu, bufor zbioru). Umożliwia to optymalne wykorzystywanie pamięci operacyjnej.

Zbiorom, których nazwy wymienione są w tym samym zwrocie SAME RECORD AREA przydzielane jest wspólne pole dokumentu. Zbiory te mogą być jednocześnie otwarte.

Zbiorom, których nazwy wymienione są w tym samym zwrocie SAME AREA przydzielone jest wspólne pole dokumentu oraz wspólny bufor zbioru. Zbiory takie nie mogą być jednocześnie otwarte.

Nazwa tego samego zbioru nie może występować w dwu różnych zwrotach SAME AREA, ani w dwu różnych zwrotach SAME RECORD AREA. Natomiast może wystąpić jednocześnie w obu tych zwrotach.

Jeśli zwrot SAME odnosi się do zbiorów, których odpowiednie obszary są różnej długości, to wspólny obszar tych zbiorów ma długość równą maksymalnej długości tych obszarów.

Zwrot SAME dotyczy zarówno zbiorów danych, jak i zbiorów sortowania. Każdy z tych zbiorów musi być opisany w części DATA.

Przykłady

SAME RECORD AREA FOR X, Y, Z

SAME AREA FOR Y, F

Zbiory Y, F, X, Z mają wspólne pole dokumentu.

Zbiory Y, F mają ponadto wspólny bufor.

Zbiory X, Z mają oddzielne bufor.

4.2.2.2.3. Zwrot MULTIPLE

Postać

MULTIPLE FILE TAPE CONTAINS [] <nazwa-zbioru-danych> []

<nazwa-zbioru-danych> jest identyfikatorem.

Zwrot ten służy do określenia, które zbiory danych znajdują się na wielozbiorowych szpulach taśmy magnetycznej.

Każdy zbiór, którego nazwa występuje w zwrocie MULTIPLE, może być związany ze szpulą taśmy magnetycznej przypisaną jednocześnie kilku

zbiorom. Nie oznacza to jednak, że wszystkie wymienione zbiory muszą znajdować się na tej samej szpuli.

Zbiór, który dzieli tę samą szpulę taśmy z innymi zbiorami można otworzyć tylko wtedy, gdy zostanie on wymieniony na liście tego zwrotu.

Każdy z wymienionych zbiorów musi być opisany w części DATA oraz musi mu być przydzielone dokładnie jedno urządzenie typu MTAPE.

Zwrot MULTIPLE dotyczy tylko zbiorów danych.

Przykład

MULTIPLE FILE TAPE TM1, TM2, TM3, TMX

Zbiory danych o nazwach TM1, TM2, TM3, TMX znajdują się na taśmach wielozbiorowych.

4.2.2.2.4. Zwrot RERUN

Postać zwrotu RERUN podana jest we wkładce.

Zwrot RERUN wskazuje, kiedy w czasie wykonania programu ma być wykonywana akcja RERUN oraz jakich zbiorów dotyczy.

Wykonanie operacji RERUN polega na zapisaniu na wyznaczonej taśmie magnetycznej stanu całej maszyny, to znaczy:

- wartości rejestrów wewnętrznych,
- zawartości całej pamięci operacyjnej oraz bębnowej,
- stanu urządzeń zewnętrznych,
- stanu licznika dokumentów i numeru ostatniego zerowatora.

Operację RERUN wykonuje się po to, aby można było w dowolnej chwili przerwać pracę programu, a następnie w dowolnej chwili wznowić wykonywanie programu od stanu, w którym wykonanie przerwano (dokładnie od stanu jaki był w chwili wykonania ostatniej operacji RERUN).

Wznówienie wykonywania programu dokonuje się za pomocą operacji RESTART (poza językiem COBOL).

O ile w zwrocie RERUN występuje opcja ON <nazwa-zbioru-1>, to stan maszyny zapisywany jest na taśmie magnetycznej związanej z tym zbiorem. Jeżeli opcja ta nie wystąpi, to stan całej maszyny zapamiętywany jest na taśmie związanej ze zbiorem <nazwa-zbioru-2> wymienionym w opcji OF. Zbiór, w którym zapamiętywany jest stan maszyny, musi być związany z taśmą magnetyczną, natomiast nie może być zbiorem sortowania, i w momencie rozpoczęcia operacji RERUN musi być otwarty dla wyjścia.

Częstotliwość wykonywania operacji RERUN definiowana jest następująco:

- zwrotem <całkowita> RECORDS; wówczas informacja o stanie maszyny zapisywana jest po każdym przesłaniu <całkowita> dokumentów z (lub do) zbioru <nazwa-zbioru-2>; zbiór taki może być zbiorem wejściowym lub wyjściowym związanym z nośnikiem papierowym, taśmą magnetyczną lub zbiorem sortowania;
- zwrotem END OF REEL; stan maszyny zapamiętywany jest bezpośrednio po wykryciu końca każdej szpuli zbioru <nazwa-zbioru-2>; taki

zbiór musi być związany z taśmą magnetyczną, ale nie może być zbiorem sortowania; może to być zbiór wejściowy lub wyjściowy;

- zwrotem RERUN MARK; stan całej maszyny wypisywany jest po wczytaniu ze zbioru <nazwa-zbioru-2> oznacznika operacji ponawiania. Zbiór ten musi być zbiorem wejściowym związanym z nośnikiem papierowym.

Jeżeli <nazwa-zbioru-2> jest zbiorem sortowania, to w zwrocie RERUN może wystąpić tylko opcja <całkowita RECORDS>. W tym przypadku musi wystąpić opcja ON <nazwa-zbioru-1>.

Po wykonaniu operacji RERUN na monitor wypisywany jest komunikat zawierający:

- identyfikator zbioru, w którym został zapisany stan maszyny (nie nazwa zbioru),
- symbol stacji tego zbioru (jedna z liter A, B, ..., F),
- numer pierwszego bloku taśmy, począwszy od którego informacja została zapisana.

Reguły określające sposób, w jaki można wznowić wykonywanie programu podano w opisie operacji RESTART.

Przykłady

RERUN ON DANE..1 EVERY 10 RECORDS OF TAPE

Operacja RERUN będzie wykonywana w trakcie opracowywania dokumentów zbioru o nazwie TAPE, po opracowaniu każdej dziesiątki dokumentów. Bloki rerunowe będą umieszczone na taśmie magnetycznej, z którą związany jest zbiór DANE..1.

RERUN EVERY END OF REEL OF ADX..14

Operacja RERUN wykonana zostanie po każdym napotkaniu końca szpuli zbioru ADX..14.

Bloki rerunowe będą umieszczone na taśmie związanej ze zbiorem ADX..14.

RERUN ON AZTM RERUN MARK OF PAP.

Operacja RERUN zostanie wykonana po każdym napotkaniu oznacznika ponawiania programu wśród dokumentów zbioru o nazwie PAP. Bloki rerunowe zostaną umieszczone na taśmie magnetycznej związanej ze zbiorem o nazwie AZTM.

Przykład części ENVIRONMENT

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SPECIAL..NAMES.

SWITCH 5 ON STATUS PIEC; OFF STATUS
IS DYM; OFF STATUS IS L2; CURRENCY SIGN
IS 'Z'.

INPUT..OUTPUT SECTION.

FILE..CONTROL.

SELECT ZB..DA ASSIGN TO MTAPE A,MTAPE B.

SELECT DANE INPUT 2.

SELECT XPRS MTAPE A, MTAPE B,MTAPE C,
MTAPE D.

I..O..CONTROL.

APPLY UNPOSITIONAL 3 RERUN MARK ':' ON DANE;
APPLY STANDARD 3 ON ZB..DA;
APPLY STANDARD ON XPRS;

SAME RECORD AREA ZB..DA, DANE;

SAME AREA FOR DANE, XPRS;

RERUN ON ZB..DA EVERY RERUN MARK OF DANE;

Objaśnienia

Wartość 0 bitu 5 trzeciego słowa nosi nazwę DYM oraz L2.

Wartość 1 bitu 5 trzeciego słowa nosi nazwę PIEC.

Niestandardowym znakiem waluty jest litera Z. Zbiór ZB..DA związany jest z taśmą magnetyczną i przypisano mu dwie stacje taśmy magnetycznej A oraz B. Zapis bloku na taśmie ma być trzykrotny.

Zbiór DANE związany jest z nośnikiem papierowym. Przypisano mu wejście symboliczne 2. Wczytywanie odbywa się techniką niepozycyjną. Część dokumentów tego zbioru ma znaczniki, a część nie (UNPOSITIONAL 3).

Znakiem rerunowym tego zbioru jest dwukropek. Zbiór XPRS związany jest z taśmą magnetyczną. Przypisano mu cztery stacje taśmy A,B,C,D. Zapis bloku na taśmie jest jednokrotny. Wspólne pole dokumentu mają zbiory ZB..DA, DANE, XPRS. Wspólny bufor mają zbiory DANE, XPRS. Zbiór ZB..DA ma oddzielny bufor.

Operacja RERUN będzie wykonywana w czasie wczytywania zbioru DANE przy każdym napotkaniu oznacznika operacji ponawiania. Bloki rerunowe będą zapisywane w zbiorze ZB..DA.

W K Ł A D K . A I

1. Postać ogólna części identyfikującej

IDENTIFICATION DIVISION

PROGRAM..ID. <nazwa-programu> . [<tekst-komentarzowy>]

2. Postać ogólna części - Urządzenia

ENVIRONMENT DIVISION.

[CONFIGURATION SECTION.

[SOURCE..COMPUTER : ZAM..41 .]
[OBJECT..COMPUTER : ZAM..41 [MEMORY SIZE <całkowita> WORDS] .]
[SPECIAL..NAMES. { <zdanie-COPY> .
[<zwrot-CURRENCY>] .
[<opis-wskaźnika>] [zwrot-CURRENCY] . } } }

[INPUT..OUTPUT SECTION.

[FILE..CONTROL.

{ <zdanie-COPY> . }
 { [deklaracja-SELECT .] }
 { zdanie-COPY }
 { zwrot-APPLY }
 { zwrot-SAME }
 { zwrot-RERUN }
 { zwrot-MULTIPLE }

I..O..CONTROL.

2.1. Postać ogólna opisu wskaźnika

SWITCH <całkowita> [ON STATUS IS <nazwa-wskaźnika>
OFF STATUS IS <nazwa-wskaźnika>]

2.2. Postać ogólna zwrotu CURRENCY

CURRENCY SIGN IS ' <znak-waluty> '

2.3. Postać ogólna deklaracji SELECT

Wersja 1

SELECT <nazwa-zbioru-danych> ASSIGN TO

$$\left\{ \begin{array}{l} \left[\text{,} \underline{\text{MTAPE}} \langle \text{litera} \rangle \right] \text{; } \underline{\text{RESERVE}} \\ \left\{ \begin{array}{l} \underline{\text{INPUT}} \\ \underline{\text{OUTPUT}} \end{array} \right\} \langle \text{całkowita} \rangle \end{array} \right\}$$

$$\left\{ \begin{array}{l} \langle \text{całkowita} \rangle \\ \underline{\text{NO}} \end{array} \right\}$$

ALTERNATE AREA

Wersja 2

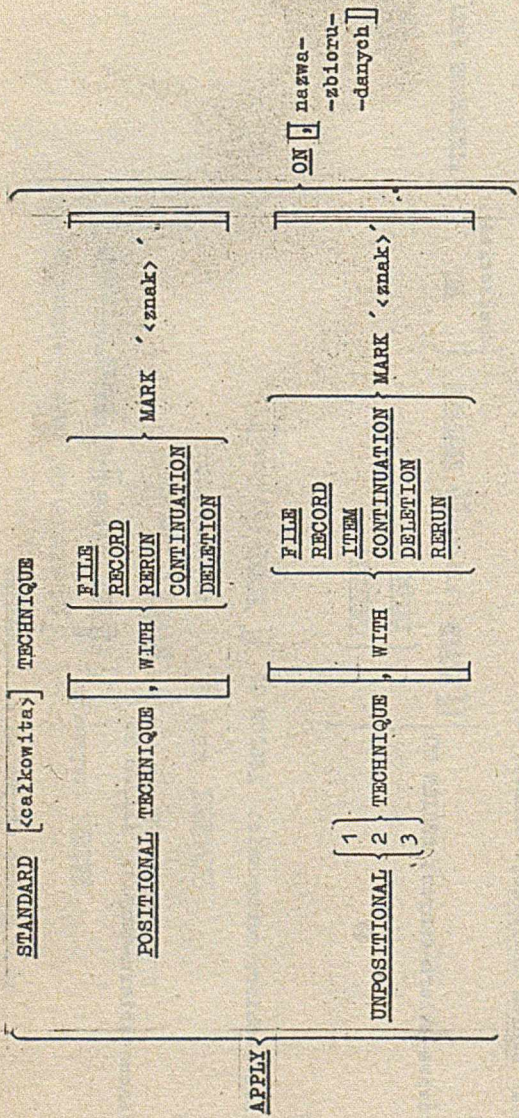
SELECT <nazwa-zbioru-sortowania> ASSIGN TO $\left[\underline{\text{MTAPE}} \langle \text{litera} \rangle \right]$

Wersja 3

SELECT <nazwa-zbioru-danych> ASSIGN TO

$$\left\{ \begin{array}{l} \underline{\text{MTAPE}} \langle \text{litera} \rangle \\ \left[\underline{\text{MTAPE}} \langle \text{litera} \rangle \right] \text{ OR } \underline{\text{MTAPE}} \langle \text{litera} \rangle \end{array} \right\}$$

2.4. Postać ogólna zwrotu APPLY



2.5. Postać ogólna zwrotu SAME

SAME [RECORD] AREA FOR [, <nazwa-zbioru>]

2.6. Postać ogólna zwrotu MULTIPLE

MULTIPLE FILE TAPE CONTAINS [, <nazwa-zbioru-danych>]

2.7. Postać ogólna zwrotu RERUN

<u>RERUN</u>	{	[<u>ON</u> <nazwa-zbioru-1> EVERY	{	<całkowita> <u>RECORDS</u>	}	OF <nazwa-zbioru-2>
			END OF <u>REEL</u>			

2.8. Postać ogólna zdania COPY

COPY <nazwa-biblioteczna>

4.3. DATA DIVISION - DANE

Postać części DATA podana jest we wkladce II. W części DATA DIVISION opisuje się wszelkiego rodzaju jednostki danych oraz jednostki robocze i związane z nimi obszary pamięci, na których działa program. Podaje się ich własności oraz strukturę. Część ta dzieli się na dwie sekcje:

- sekcję zbiorów FILE SECTION; zawiera ona opis zbiorów danych oraz zbiorów sortowania;
- sekcję roboczych WORKING..STORAGE SECTION; zawiera ona opis miejsc roboczych.

4.3.1. FILE SECTION - SEKCJA ZBIORÓW

W sekcji tej opisuje się strukturę i własności poszczególnych zbiorów danych wczytywanych do pamięci oraz wypisywanych z niej, jak również przydziela się pola pamięci odpowiadające tym zbiorom. W sekcji zbiorów opisuje się również obszary używane do operacji sortowania. Ze względu na to, że odnoszące się do nich opisy mają strukturę i charakter analogiczny jak opisy zbiorów danych, wprowadza się fikcyjne pojęcie zwane zbiorem sortowania.

Zbiorów danych i zbiorów sortowania może być wiele. Każdy opisywany zbiór musi zawierać opis co najmniej jednego dokumentu.

4.3.1.1. FD - opis zbioru danych

Postać

FD nazwa-zbioru-danych

;	zwrot-DATA
;	zwrot-LABEL
[;	zwrot-VALUE OF]
[;	zwrot-BLOCK]
[;	zwrot-RECORD]

<nazwa-zbioru-danych> jest identyfikatorem.

Opis zbioru danych obejmuje rezerwację pola pamięci zwanego buforem zbioru (zwrot-BLOCK), metryk zbioru (zwroty LABEL oraz VALUE OF) oraz zawiera listę nazw dokumentów tego zbioru (zwrot-DATA). Zwroty LABEL oraz DATA muszą wystąpić w opisie zbioru.

Kolejność występowania zwrotów w opisie jest dowolna. W razie powtórnego wystąpienia któregoś z nich brane pod uwagę jest pierwsze wystąpienie zwrotu w opisie.

Znaczenie poszczególnych zwrotów omówione jest dalej.

4.3.1.2. SD - opis zbioru sortowania

Postać

$$SD \langle \text{nazwa-zbioru-sortowania} \rangle \left[\begin{array}{l} ; \text{zwrot-DATA} \\ ; \text{zwrot-BLOCK} \\ ; \text{zwrot-RECORD} \end{array} \right]$$

<nazwa-zbioru-sortowania> jest identyfikatorem.

Operacja sortowania wymaga określonego zestawu taśm magnetycznych, na których odbywa się sortowanie, bufora pamięci dla tych taśm oraz pola pamięci, z którego pobierane są dokumenty do sortowania oraz do którego wpisywane są kolejne dokumenty po sortowaniu. Z tego względu wygodnie jest traktować ten zestaw obiektów jako zbiór. Zbiór taki nosi nazwę zbioru sortowania. Bufor pamięci dla taśm sortowania nazywa się buforem zbioru sortowania, a pole, w którym umieszcza się dokumenty - polem dokumentu. Zbiór sortowania nie posiada żadnych me-

tryk ani dokumentów specjalnych. Do zbioru sortowania nie mogą odnosić się zwroty i instrukcje dotyczące metryk ani też operacji otwierania i zamykania zbiorów.

Opis zbioru sortowania obejmuje rezerwację bufora zbioru (zwrot-BLOCK) oraz zawiera listę dokumentów tego zbioru.

Wielkość pola dokumentu, bufora zbioru oraz bloku informacji na taśmie magnetycznej określa się analogicznie jak dla zbioru danych z tą różnicą, że długość bufora zbioru sortowania równa jest długości bloku taśm sortowania pomnożonej przez liczbę taśm sortowania. Postać zwrotów występujących w opisie zbioru sortowania i dotyczące ich reguły, z wyjątkiem długości bufora, są takie same jak dla zbioru danych.

Opis dokumentu dla tego zbioru jest identyczny jak dla zbioru danych.

4.3.1.3. Zwrot DATA

Postać

DATA RECORD IS [nazwa dokumentu]

Na liście zwrotu DATA muszą być wymienione nazwy wszystkich dokumentów występujących w opisywanym zbiorze danych.

Przykład

DATA RECORD IS AX2, B..Q, D2T

Opisywany zbiór składa się z trzech typów dokumentów o nazwach AX2, B..Q, D2T.

4.3.1.4. Zwrot LABEL

Postać

LABEL RECORD IS $\left\{ \begin{array}{l} \text{OMITTED} \\ \text{STANDARD WITH } [\text{GENER}] \end{array} \right\}$

Zwrot LABEL określa, czy zbiór jest metrykowany, czy nie. Wersja OMITTED oznacza, że zbiór nie jest metrykowany. Fakt ten ma różne znaczenie zależnie od rodzaju zbioru.

Dla zbiorów związanych z nośnikiem papierowy oznacza to, że zbiór nie ma metryki początku zbioru (taki zbiór innych metryk w ogóle nie posiada).

Zbiory związane z taśmą magnetyczną zawsze posiadają metryki (początku i końca zbioru). Dla zbiorów tych własność OMITTED oznacza, że w ramach wykonania instrukcji OPEN INPUT oraz CLOSE pozycje metrykowe nie są sprawdzane, mimo że metryka zostaje wczytana.

Ponadto wersja OMITTED oznacza, że dla tego zbioru nie zostaną wykonane niestandardowe procedury metrykowe określone w zdaniu USE.

Wersja STANDARD określa, że zbiór ma standardowe metryki.

Wyrażenie WITH GENER może wystąpić tylko dla zbiorów taśmowych. Użycie tego wyrażenia oznacza, że w trakcie otwarcia tego zbioru (instrukcja OPEN) na monitorze należy podać numer generacji zbioru. O ile wyrażenie WITH GENER nie wystąpi, to numer ten ma wartość 1.

Zwrot LABEL nie może wystąpić w opisie zbioru sortowania ponieważ zbiór ten nie posiada żadnych metryk.

Przykład

LABEL RECORD IS OMITTED

Zbiór nie jest metrykowany.

4.3.1.5. Zwrot VALUE OF

Postać

<u>VALUE OF</u>	,	<u>IDENT</u> <tekst>
		<u>SECT</u> <całkowita>
		<u>SECUR</u> <całkowita>

Zwrot ten określa wartości pewnych pozycji metrykowych zbioru. Pozycje te są następujące:

- IDENT - identyfikator zbioru; wartością tej pozycji może być tekst nie dłuższy niż 18 znaków;
- SECT - numer sekcji; sekcja jest to dla zbiorów wieloszpulowych część zbioru mieszcząca się na jednej szpuli; zbiory jednoszpulowe tworzą jedną sekcję; wartością tej pozycji może być liczba całkowita krótka; jest to numer pierwszej z kolei sekcji zbioru;
- SECUR - liczba dni ważności zbioru; wartością tej pozycji może być liczba całkowita krótka.

O ile pozycjom tym wartości nie zostaną nadane to przyjmuje się:

IDENT - nazwę zbioru (występującą za znacznikiem FD) obciętą lub uzupełnioną spacjami do 18 znaków,

SECT - całkowita 1,

SECUR - całkowita 0.

Jeżeli któraś opcja wystąpi w zwrocie VALUE OF więcej niż raz, to pod uwagę bierze się jej pierwsze wystąpienie.

Dla zbiorów związanych z nośnikiem papierowym może wystąpić tylko zmienna IDENT. Pozostałe odnoszą się tylko do zbiorów związanych z taśmą magnetyczną.

Zwrot VALUE OF nie może wystąpić w opisie zbioru sortowania ponieważ zbiór ten żadnych metryk nie posiada.

Przykład

VALUE OF IDENT 'PRZEDADRESOWANIA', SECT 3,
SECUR 120

Identyfikatorem zbioru jest tekst uzupełniony trzema spacjami

PRZEDADRESOWANIA

Wartości numerów sekcji zaczynają się od liczby 3.

Zbiór ważny jest 120 dni.

4.3.1.6. Zwrot BLOCK

Postać

$$\underline{\text{BLOCK}} \text{ CONTAINS } \langle \text{całkowita} \rangle \left\{ \begin{array}{l} \underline{\text{RECORDS}} \\ \underline{\text{CHARACTERS}} \end{array} \right\}$$

gdzie $\langle \text{całkowita} \rangle$ musi być liczbą całkowitą bez znaku.

Zwrot BLOCK określa wielkość bufora opisywanego zbioru, a dla zbiorów związanych z taśmą magnetyczną również długość bloku informacji na taśmie.

Wersja RECORDS oznacza, że długość bufora liczona jest w dokumentach.

Wersja CHARACTERS oznacza, że długość bufora liczona jest w znakach (słowach pamięci).

Wielkość $\langle \text{całkowita} \rangle$ określa ile dokumentów lub znaków zawiera bufor.

Dla zbiorów pozycyjnych zwrot ten jest pomijany.

Dla zbiorów niepozycyjnych wolno używać jedynie wersji CHARACTERS.

Dla zbiorów taśmowych wolno używać obydwu wersji.

Jeżeli zwrot ten nie wystąpi, a zbiór wymaga buforu, to bufor przydzielany jest automatycznie, a jego długość równa jest długości pola dokumentu.

Przykład

BLOCK 230

Bufor zbioru zajmuje 230 słów maszyny.

4.3.1.7. Zwrot RECORD

Postać

RECORD CONTAINS <całkowita> CHARACTERS

Zwrot ten jest przez translator ignorowany; w innych implementacjach służy on do określania długości pola dokumentu. W tej implementacji długość pola dokumentu określana jest przez translator automatycznie.

Przykład

RECORD 5 CHARACTERS

4.3.1.8. Opis dokumentu

Postać podana została we wkładce II.

Opis dokumentu określa porządek i hierarchię pozycji tego dokumentu oraz własności poszczególnych pozycji.

Nazwy pozycji oraz dokumentu są identyfikatorami.

Nazwy różnych pozycji oraz dokumentów mogą być jednakowe ze względu na to, że można stosować tak zwaną kwalifikację, to znaczy w części PROCEDURE wraz z nazwą pozycji podawać nazwy pozycji obejmujących (nie koniecznie bezpośrednio) aż do nazwy zbioru włącznie. Nazwy muszą być tak dobrane, aby stosując odpowiednie kwalifikacje można było odwoływać się do poszczególnych pozycji w sposób jednoznaczny.

<numer-poziomu> jest całkowitą, przyjmującą wartości od 2 do 49. Numer ten określa hierarchię poszczególnych pozycji w dokumencie. Pozycja o numerze poziomu n zawarta jest w najbliższej poprzedzającej ją pozycji o numerze $q < n$. Pozycja nie zawierająca innych pozycji jest pozycją elementarną. Pozycja zawierająca inne pozycje nosi nazwę grupowej.

Nazwę FILLER używać można, gdy w programie nie występuje odwołanie do danej pozycji. Jednak długość i inne własności tej pozycji muszą być opisane.

Znaczenie i budowę poszczególnych zwrotów podana jest poniżej.

Zwroty FIXED, FIXED2, FLOAT

Pozycje opisane tymi zwrotami nazywają się obliczeniowymi.

1) W sekcji zbiorów zwroty te mogą wystąpić w opisie następujących pozycji elementarnych:

- należących do zbiorów taśmowych,
- należących do zbiorów niepozycyjnych,
- należących do dowolnych zbiorów, o ile pozycje te opisane są ponadto zwrotem REDEFINES,
- należących do zbiorów sortowania.

Zwroty te nie mogą wystąpić w opisie pozycji zbiorów pozycyjnych (chyba że pozycje te są opisane zwrotem REDEFINES).

2) W sekcji roboczych zwroty te mogą wystąpić w opisie dowolnej pozycji elementarnej.

4.3.1.8.1. Zwrot FIXED

Postać

FIXED

Zwrot ten określa, że wartościami opisywanej pozycji mogą być tylko liczby całkowite krótkie w postaci binarnej. Polem takiej pozycji jest jedno miejsce pamięci. Zawiera ono:

w bicie 0 znak algebraiczny liczby,
w bitach 1-23 wartość bezwzględną liczby,
binarnie.

4.3.1.8.2. Zwrot FIXED2

Postać

FIXED2

Zwrot ten określa, że wartościami opisywanej pozycji mogą być tylko liczby całkowite długie w postaci binarnej. Polem takiej pozycji są dwa kolejne miejsca (słowa) pamięci.

Zawierają one:

w bitach 0 i 24 znak algebraiczny liczby,
w bitach 1-23 i 25-47 wartość bezwzględną
liczby, binarnie.

4.3.1.8.3. Zwrot FLOAT

Postać

FLOAT

Zwrot ten określa, że wartością opisywanej pozycji mogą być tylko liczby zmiennoprzecinkowe

w postaci binarnej. Pole takiej pozycji zajmuje dwa kolejne miejsca (słowa) pamięci. Zawierają one:

- w bicie 0 znak algebraiczny mantysy,
- w bicie 24 znak algebraiczny cechy,
- w bitach 1-23 i 25-39 wartość bezwzględną mantysy, binarnie,
- w bitach 40-47 wartość bezwzględną cechy, binarnie.

4.3.1.8.4. Zwrot PICTURE

Postać

$$\left\{ \begin{array}{l} \text{PICTURE} \\ \text{PIC} \end{array} \right\} \text{ IS } \langle \text{obraz} \rangle$$

Zwrot PICTURE służy do opisywania postaci pozycji elementarnych. Zawartość pozycji opisanej tym zwrotem przedstawiana jest w polu pozycji jako ciąg znaków (w kodzie KWB). Każdy znak takiej pozycji zajmuje jedno słowo pamięci i zapisany jest w ośmiu najmniej znaczących bitach tego słowa. Polem pozycji jest ciąg złożony z tylu kolejnych słów pamięci, ile w obrazie jest symboli reprezentujących znak. Pozycje opisane zwrotem PICTURE nazywają się pozycjami znakowymi.

Zwrot PICTURE może wystąpić w opisie pozycji dowolnych zbiorów, ale tylko pozycji elementarnych.

Zwrotem tym muszą być opisane wszystkie pozycje elementarne zbiorów pozycyjnych.

Wielkość <obraz> jest ujętym w apostrofy ciągiem co najwyżej 30 znaków, tzw. symboli obrazu, zbudowanym według poniżej omówionych reguł.

Zbiór symboli obrazu jest następujący:

A X 9 V P . , S Z W * + - TR CR DB O B

Wyróżniamy cztery rodzaje pozycji znakowych:

- tekstowe,
- liczbowe,
- tekstowe wydawnicze,
- liczbowe wydawnicze.

Pozycja tekstowa

Pozycja tekstowa może zawierać dowolny ciąg znaków, którego długość nie przekracza długości pola tej pozycji.

Obraz opisujący pozycję tekstową (tzw. obraz tekstowy) może być zbudowany z symboli tekstowych

A X 9

z tym, że musi zawierać co najmniej jeden symbol A lub X.

Każdy z symboli A, X, 9 reprezentuje dowolny znak kodu KWS i powoduje zarezerwowanie jednego miejsca w polu pozycji.

Pozycja liczbowa

Pozycja liczbowa może zawierać ciąg cyfr nie zawierający kropki dziesiętnej, ewentualnie poprzedzony znakiem algebraicznym plus lub minus.

W polu pozycji liczbowej może być umieszczonych co najwyżej 18 cyfr.

Obraz opisujący pozycję liczbową (obraz liczbowy) może być zbudowany z symboli liczbowych

9 S V P

Znaczenie tych symboli jest następujące:

9 - oznacza miejsce pola pozycji, do którego może być wstawiona tylko cyfra,

S - oznacza, że liczba na nośniku może być poprzedzona znakiem plus lub minus. Jeżeli w obrazie wystąpi symbol S, to musi on być pierwszym symbolem tego obrazu.

Symbole 9 i S są wliczane do długości pozycji i każdy z nich powoduje zarezerwowanie jednego miejsca w polu pozycji.

V - Symbol ten swoim położeniem wyznacza położenie tzw. domyślnej kropki dziesiętnej, to znaczy miejsca, w którym kończy się część całkowita, a zaczyna część ułamkowa liczby.

Symbol V nie reprezentuje znaku pola pozycji i nie jest wliczany do długości jej pola. Może on wystąpić w tym samym obrazie tylko raz i nie może występować jednocześnie z symbolem P.

P - Symbol P służy do określania położenia domyślnej kropki dziesiętnej, gdy leży ona poza polem pozycji. Reprezentuje on te znaki cyfrowe opisywanej pozycji, dla których nie jest rezerwowane miejsce w polu pozycji. Symbol ten może występo-

wać w obrazie pojedynczo lub jako ciąg bezpośrednio po sobie następujących symboli P. Taki ciąg może wystąpić w obrazie tylko raz przed lub po ciągu symboli cyfrowych 9. Domyslna kropka dziesiętna znajduje się przed pierwszym symbolem P, gdy ciąg poprzedza miejsca cyfrowe lub za ostatnim; gdy ciąg symboli P wystąpił na końcu obrazu. Symbol P nie reprezentuje znaku pola i nie jest wliczany do długości pozycji. Nie może on występować w tym samym obrazie jednocześnie z symbolem V.

Pozycja tekstowa wydawnicza

Pozycja tekstowa wydawnicza może zawierać dowolny tekst (taki jak pozycja tekstowa) wzbogacony o znaki wydawnicze:

- spację,
- zero.

Znaki wydawnicze mogą dowolnie (zgodnie z obrazem) rozdzielać znaki tekstu. Liczba znaków tekstu nie może przekraczać liczby symboli A X 9 umieszczonych w obrazie.

Obraz pozycji tekstowej wydawniczej (obraz tekstowy wydawniczy) może być zbudowany:

- z symboli tekstowych A X 9,
- z symboli wydawniczych B 0 (zero).

Obraz tekstowy wydawniczy może być ciągiem o dowolnej konfiguracji tych symboli z tym, że musi on zawierać co najmniej jeden z symboli A lub X oraz co najmniej jeden symbol wydawniczy.

Znaczenie symboli tekstowych A X 9 jest analogiczne jak dla obrazu tekstowego.

Znaczenie symboli wydawniczych:

B - * reprezentuje miejsce pola, do którego jest wstawiana spacja,

O - reprezentuje miejsce pola, do którego jest wstawiana cyfra zero.

Każdy z symboli obrazu tekstowego wydawniczego powoduje zarezerwowanie jednego miejsca w polu pozycji.

Pozycja liczbowa wydawnicza

Pozycja liczbowa wydawnicza może zawierać ciąg cyfr zawierający kropkę dziesiętną poprzedzony znakiem algebraicznym, znakiem waluty lub zakończony znakiem algebraicznym lub jednym z symboli CR DB.

CR oznacza kredyt,

DB oznacza debet.

Ponadto w ciągu tym mogą wystąpić spacje i/lub przecinki. Znaki pozycji liczbowej wydawniczej rozmieszczone są zgodnie z obrazem takiej pozycji. W polu pozycji liczbowej wydawniczej może być umieszczonych co najwyżej 18 cyfr niezależnie od liczby znaków nie cyfrowych.

Obraz pozycji liczbowej wydawniczej (obraz liczbowy wydawniczy) może być zbudowany z symboli:

- liczbowych 9 P V,
- wydawniczych Z + - . , * W T R CR
DB B O

ale musi zawierać co najmniej jeden symbol wydawniczy.

Obraz taki musi być zbudowany według reguł podanych w tablicy kontekstów symboli obrazu.

Znaczenie i sposób stosowania symboli obrazu liczbowego wydawniczego jest następujące:

- | | | |
|-----------|--|--|
| 9 P V | | symbole te mają takie same znaczenie jak w obrazie liczbowym, |
| Z W + - * | | symbole te są związane z operacją supresji (zamiany) prowadzących zer (tzw. symbole supresji). Prowadzące zero jest to zero zaczynające część całkowitą liczby, to znaczy występujące przed jej pierwszą cyfrą niezerową oraz przed kropką dziesiętną. Operacja supresji polega na zastąpieniu prowadzącego zera określonym znakiem. Symbole Z W + - * mogą występować w obrazie pojedynczo lub jako ciąg takich samych symboli. Ciąg taki może być co najwyżej rozdzielony niektórymi symbolami wstawiania (B, O). Omawiane symbole mogą reprezentować następujące znaki cyfrowe: |
| | | - początkowe lub wszystkie znaki cyfrowe części całkowitej, |
| | | - wszystkie znaki cyfrowe liczby, a ponadto znak waluty, gwiazdka lub algebraiczny znak liczby, poprzedzające znaki cyfrowe. |

Wystąpienie każdego z tych symboli powoduje zarezerwowanie jednego miejsca w polu pozycji.

- * Z Reprezentuje miejsce pola pozycji, do którego wstawiony może być znak cyfry z tym, że jeśli ta cyfra jest zerem prowadzącym, to podlega ona operacji supresji odpowiednim znakiem:

dla symbolu Z spacją,

dla symbolu * gwiazdką.

- W + - Znaczenie tych symboli zależy od tego, czy występują pojedynczo czy też jako ciąg.

Jeśli symbol występuje pojedynczo, to reprezentuje jedno miejsce pola pozycji, do którego wstawiany jest następujący znak odpowiednio dla każdego symbolu:

- W Znak waluty zdefiniowany zwrotem CURRENCY SIGN w paragrafie SPECIAL..NAMES lub litera W, gdy zwrot ten nie wystąpił,
- + algebraiczny znak liczby zgodny z jej wartością,
- dla liczby dodatniej wstawiana jest spacja, a dla ujemnej znak minus.

Jeśli omawiany symbol wystąpi jako ciąg n jednakowych symboli to reprezentuje on:

- n - 1 cyfrowych znaków pola pozycji podlegających supresji spacją i
- 1 miejsce pola pozycji, do którego wstawiany jest algebraiczny znak liczby lub znak waluty.

Jeśli wartość liczby jest taka, że supresja zer ma miejsce, to znak algebraiczny lub znak waluty umieszczany jest w polu pozycji bezpośrednio przed pierwszą cyfrą znaczącą lub przed kropką dziesiętną.

Jeżeli wszystkie znaki cyfrowe pola pozycji opisane są symbolami supresji i do takiego pola przesłana zostanie liczba równa zero, to do każdego miejsca pola pozycji wstawiana jest spacja. Zastępowanie znaku spacją dotyczy wówczas także miejsc zarezerwowanych na znak algebraiczny, znak waluty oraz na inne znaki wydawnicze umieszczone wewnątrz ciągu symboli supresji w obrazie.

O B - , . RT CR DB są to symbole wstawiania. Odpowiedni znak dla danego symbolu wstawiany jest w tym miejscu pola, w którym symbol wystąpił w obrazie.

Symbole CR i DB zajmują po dwa miejsca w polu pozycji, a pozostałe jedno miejsce.

Znaczenie i sposób stosowania poszczególnych symboli wstawiania jest następujące:

O B , symbole te oznaczają:

- symbol B spację,
- symbol O zero,
- symbol , przecinek

i mogą wystąpić w dowolnym miejscu obrazu dowolną liczbę razy z zastrzeżeniem nieprzekroczenia długości obrazu.

Jeżeli symbol taki wystąpi wewnątrz ciągu symboli supresji to znak, który reprezentuje, również podlega supresji wraz z otaczającymi go cyframi, (to znaczy zostanie zastąpiony spacją lub gwiazdką.).

Symbol ten reprezentuje znak kropki dziesiętnej, która wstawiana jest do pola pozycji. Kropka może wystąpić w obrazie tylko raz i nie może występować jednocześnie z symbolami V i P.

Symbol kropki dziesiętnej może wystąpić wewnątrz ciągu symboli supresji tylko wtedy, gdy wszystkie znaki cyfrowe liczby opisane są tymi symbolami. Wówczas, o ile liczba posyłana do pola tej pozycji ma wartość zero, to kropka dziesiętna zostaje zastąpiona spacją.

T R CR DB Symbole te mogą wystąpić w obrazie tylko pojedynczo na końcu obrazu i nie mogą występować jednocześnie w tym samym obrazie, ani też w obrazie, w którym wystąpiły symbole + lub -. Reprezentują one jedno (dla T i R) oraz dwa (dla CR i DB) miejsca w polu pozycji, do których wstawiane są odpowiednie znaki w zależności od znaku algebraicznego liczby. Obrazuje to następująca tabelka:

Symbol obrazu	Znak wstawiany do pola	
	pozycja dodatnia	pozycja ujemna
T	+	-
R	spacja	-
CR	2 spacje	CR
DB	2 spacje	DB

Uwaga

Po każdym symbolu obrazu, który może być w nim powtarzany wielokrotnie, może występować liczba całkowita bez znaku, ujęta w nawiasy

okrągłe. Liczba ta oznacza ilość powtórzeń symbolu, po którym następuje.

Pozycje liczbowe oraz pozycje liczbowe wydawnicze mogą zawierać co najwyżej 18 cyfr znaczących. Należy jednak zauważyć, że jeżeli pozycja taka ma zostać przesłana do pola innej pozycji typu obliczeniowego, to jej moduł nie może przekraczać maksymalnej wartości modułu liczby przewidzianej dla tych typów, to znaczy dla pozycji:

- typu FIXED $2^{23}-1$ czyli 8 388 607,
- typu FIXED2 $2^{46}-1$ czyli 70 368 744 177 663,
- typu FLOAT co najwyżej 11 cyfr znaczących.

Dozwolone konteksty symboli obrazów ilustruje tabela zamieszczona na następnej stronie.

Tabela ta wskazuje jakie symbole mogą wystąpić w tym samym obrazie jednocześnie oraz jaka jest dozwolona ich kolejność.

Ponadto tabela podaje informację o tym, w jakim rodzaju obrazu może wystąpić każdy z symboli oraz ile każdy z nich zajmuje miejsc w polu pozycji.

Znak + w tabeli wskazuje, że w obrazie po symbolu umieszczonym w wierszu "poprzednik" - może wystąpić w dalszej części obrazu (nie koniecznie bezpośrednio za nim) symbol umieszczony w kolumnie "następnik". Znak + w tabeli wskazuje również, że dany symbol może wystąpić w określonym rodzaju obrazu. Cyfra w tabeli wskazuje ile miejsc w polu pozycji zajmuje odpowiedni symbol.

4.3.1.8.5. Zwrot BLANK

Postać

BLANK WHEN ZERO

Zwrot BLANK powoduje, że do pola pozycji opisanej tym zwrotem posyłane są spacje w przypadku zerowej wartości tej pozycji. Zwrot ten dotyczy tylko pozycji elementarnych określonych jako liczbowe wydawnicze. Zwrot BLANK pomijamy wówczas, gdy w obrazie pozycji wystąpi symbol * lub też gdy zwrot ten wystąpi w opisie pozycji innej niż liczbowa wydawnicza. Zwrot ten nie spacjuje prowadzących zer, gdy wartością liczby nie jest zero.

4.3.1.8.6. Zwrot JUSTIFIED

Postać

JUSTIFIED RIGHT

Zwrot JUSTIFIED powoduje niestandardowy sposób przesyłania danych opisywanej pozycji tekstowej. Może on wystąpić tylko w opisie elementarnych pozycji tekstowych i tekstowych wydawniczych.

Standardowe przesyłanie tekstu odbywa się następująco: pierwszy z lewej znak umieszczany jest w pierwszym od lewej miejscu pola pozycji docelowej, drugi w drugim od lewej, i tak dalej; gdy przesyłany tekst jest dłuższy niż pole pozycji docelowej wówczas skrajne prawe znaki są obcinane; gdy przesyłany tekst jest krótszy, wówczas uzupełnia się go spacjami z prawej strony.

Jeśli pozycja docelowa opisana jest zwrotem JUSTIFIED, to przesyłanie kolejnych znaków odbywa się w kolejności odwrotnej: pierwszy z prawej znak posyłany jest do pierwszego z prawej miejsca pola, drugi z prawej do drugiego z prawej itd. Ewentualne obcinanie przesyłanego tekstu lub uzupełnianie go spacjami wykonywane jest od jego lewego końca.

4.3.1.8.7. Zwrot VALUE

Postać

VALUE IS <stała>

Zwrot VALUE służy do nadawania wartości początkowej pozycjom w sekcji roboczych (to znaczy określa zawartość pola pozycji) lub do nadawania wartości znacznikom w chwili startu programu w dokumentach wejściowych. Zwrot ten brany jest pod uwagę jedynie w sekcji roboczych i w opisie dokumentów zbiorów wejściowych wczytywanych techniką niepozycyjną; w pozostałych przypadkach jest pomijany.

Zwrot VALUE może wystąpić tylko w opisie pozycji elementarnych, ale jeżeli pozycja elementarna opisana jest ponadto zwrotem REDEFINES lub OCCURS; to również jest pomijany.

Wielkość <stała> musi być tego samego typu co opisywana pozycja, to znaczy musi być:

- liczbą albo liczbową stałą figuratywną ZERO dla pozycji obliczeniowych oraz znakowych liczbowych i liczbowych wydawniczych,
- tekstem albo tekstową stałą figuratywną SPACE, LOW..VALUE, HIGH..VALUE dla pozycji

znakowych tekstowych i tekstowych wydawniczych.

Wówczas określana wartość początkowa pozycji musi zawierać dokładnie tyle znaków, ile w jej obrazie występuje symboli reprezentujących miejsca cyfrowe lub tekstowe.

Znacznik dokumentu musi być pozycją znakową. Służy on do rozpoznawania typu dokumentu zbiorów niepozycyjnych wczytywanych techniką 2 i 3.

4.3.1.8.8. Zwrot OCCURS

Postać

OCCURS <całkowita> TIMES

Zwrot OCCURS określa, że opisywana pozycja jest macierzą oraz wyznacza liczbę składowych tej macierzy. Może on występować na dowolnych poziomach z wyjątkiem 01 i 88.

Wielkość <całkowita> jest wymiarem macierzy i określa ilość składowych macierzy. <całkowita> musi być liczbą całkowitą bez znaku.

Pozycje opisane zwrotem OCCURS mogą się dowolnie zagnieżdżać, to znaczy pozycja opisana zwrotem OCCURS może zawierać pozycje również opisane zwrotem OCCURS.

Do pozycji będących składowymi macierzy wolno odwoływać się, w części PROCEDURE DIVISION, jedynie przez zmienną indeksowaną. Indeks może się zmieniać w zakresie od 1 do <całkowita> lub do iloczynu odpowiednich wielkości <całkowita>.

3.3.1.8.9. Zwrot REDEFINES

Postać

REDEFINES <nazwa-danej>

gdzie <nazwa danej> może być kwalifikowana, ale nie może być indeksowana.

Zwrot ten umożliwia opisywanie tego samego pola pamięci na różne sposoby. Pozycja opisana zwrotem REDEFINES zajmuje ten sam obszar pamięci w polu dokumentu, co pozycja wskazana przez <nazwę danej>. Obszar ten otrzymuje tylko nowe własności. Pole pozycji <nazwa-danej> nie może być mniejsze niż pole pozycji opisanej zwrotem REDEFINES.

Opis pozycji <nazwa-danej> musi poprzedzać opis pozycji, w którym występuje zwrot REDEFINES.

Pozycja <nazwa-danej> nie może być opisana zwrotem REDEFINES.

Zwrot REDEFINES nie może występować w opisie pozycji o numerze 88, a w sekcji zbiorów w opisie dokumentu (poziom 01).

W sekcji roboczych zwrot ten może wystąpić na poziomie 01.

W danych tworzących dokument zbioru wejściowego nie mogą występować wartości dla pozycji opisanych zwrotem REDEFINES.

Zwrot REDEFINES występujący w opisie pozycji musi wystąpić w tym opisie bezpośrednio po nazwie danej pozycji.

4.3.1.8.10. Deklaracja nazwy warunku

Postać

88<identyfikator>; VALUE IS $\left[\left[\text{stała} \left[\text{THRU} \langle \text{stała} \rangle \right] \right] \right]$

gdzie <identyfikator> jest nazwą warunku.

Deklaracja nazwy warunku definiuje przedziały wartości pozycji elementarnej oraz nadaje im nazwę, tzw. nazwę warunku.

Deklaracja nazwy warunku dotyczy najbliższej poprzedzającej ją pozycji elementarnej. Pozycja elementarna może mieć kilka deklaracji warunku.

Pozycja, której dotyczy deklaracja nazwy warunku nazywa się zmienną warunkową.

Typ każdej <stałej> musi zgadzać się z typem zmiennej warunkowej. Zgodność długości, skali itp. nie jest konieczna.

Nazwa warunku przyjmuje wartość "prawda", gdy zmienna warunkowa przyjmuje wartość z przedziału określonego dla danej nazwy warunku, a wartość "fałsz" w przeciwnym przypadku.

Nazwy warunku można używać wyłącznie jako warunku, to znaczy napisanie w zdaniu warunkowym pewnej nazwy warunku równoważne jest napisaniu warunku badającego czy wartość zmiennej warunkowej mieści się w przedziale wartości nazwanego tą nazwą warunku.

Przykład

W DATA DIVISION występuje taki opis pozycji

⋮

03 ALA; FIXED.

88 A; VALUE IS +5 .

⋮

W części PROCEDURE DIVISION równoważne są takie zdania:

IF ALA EQUAL +5 THEN...

ta część może być zastąpiona nazwą warunku

IF A THEN...

4.3.2. WORKING..STORAGE SECTION - SEKCJA ROBOCZYCH

Postać

WORKING..STORAGE SECTION .

[[<opis-pojedynczej-pozycji>]]

[[<opis-dokumentu-roboczego>]]

Opis pojedynczej pozycji ma taką samą postać jak opis pozycji elementarnej w sekcji zbiorów z tym, że numer poziomu takiej pozycji równy jest 77.

Opis dokumentu roboczego jest analogiczny do opisu dokumentu zbioru danych z tą różnicą, że w dokumencie roboczym na poziomie 01 może wystąpić zwrot REDEFINES. Zwrot ten, o ile wystąpi, musi następować bezpośrednio po nazwie dokumentu. Znaczenie poszczególnych zwrotów opisu dokumentu oraz pozycji jest takie same jak dla zbiorów danych oraz sortowania.

Postać opisu dokumentu roboczego oraz pozycji pojedynczej w sekcji roboczych podano we wkładce II.

Przykład

DATA DIVISION.

FILE SECTION.

FD ZB..DA; DATA RECORD WX, PKR, ZOJ;
LABEL RECORD IS STANDARD; BLOCK 2000;
VALUE OF IDENT 'PRZEMIESZCZENIA..7'.

01 WX.

02 A; FIXED.

02 B; OCCURS 3.

03 DD; PIC '9(7)'.

03 KK; PIC 'AAA'.

03 WK; FLOAT.

02 Q; FIXED2.

01 PKR; PICTURE 'A(20)'; JUSTIFIED RIGHT.

01 ZOJ.

02 X; PIC 'S999V99'.

02 F; FLOAT; OCCURS 4.

FD DANE; LABEL RECORD OMITTED; BLOCK 5 RECORDS;
DATA RECORD DV1, DV2, XQ, A45.

01 DV1.

02 Q; PIC '9'; VALUE 4.

02 R; FLOAT.

02 WG; PIC 'A(30)'.

01 DV2.

02 R; FIXED.

02 MA; OCCURS 5.

03 G; PIC 'A(2)'.

03 F; PIC '99PPP'.

03 H; PIC '99.999'; BLANK WHEN ZERO.

02 MAX; REDEFINES MA.

03 A; FIXED.

03 B; OCCURS 3.

04 D; FLOAT.
04 F; FLOAT.
03 G.
04 T; FIXED.
88 T1; VALUE IS 0 THRU 100.
88 T2; VALUE IS -100 THRU-1.
03 W; OCCURS 10; FIXED.
01 A45; PIC '9'; VALUE 7.
01 XQ.
02 LI; OCCURS 16; FIXED2.

SD XPRS; DATA RECORD WBT.
01 WBT.
02 S1; FIXED2.
02 S2; FIXED2.
02 S3.
03 FILLER; PIC '9'.
03 FILLER; FLOAT.

WORKING..STORAGE SECTION.

77 ROB1; FIXED; OCCURS 12.
77 ST; FLOAT; VALUE IS 1.3.
01 DOK.
02 P1; PIC '****.*'.
02 DR; FLOAT.
01 POL.
02 P; PIC 'S99'
02 K; FLOAT.
01 DOX; REDEFINES DOK.
02 FILLER; OCCURS 9; FIXED.

Objaśnienia

W przykładzie tym występują te same zbiory, które zostały opisane w przykładzie w części ENVIRONMENT. Z opisu zbioru ZB..DA wyni-

ka, że zbiór ten składa się z trzech dokumentów: WX, PKR, ZOJ. Metryki tego zbioru są standardowe. Identyfikatorem zbioru jest tekst PRZEMIESZCZENIA..7. Bufor tego zbioru ma 2000 słów.

Z opisów dokumentów wynika, że długość pola dokumentu wynosi 39 słów.

Długość poszczególnych dokumentów jest następująca

$$WX \quad 1+3*(7+3+2)+2 = 39 \text{ słów}$$

$$PKR \quad 20 \text{ słów}$$

$$ZOJ \quad 6 + 4 * 2 = 14 \text{ słów}$$

Zbiór DANE zdefiniowany jest następująco:

zbiór metryk nie posiada; bufor zbioru ma długość 5 dokumentów czyli 255 słów; w skład zbioru wchodzi 4 typy dokumentów. Długości poszczególnych dokumentów są następujące:

$$DV1 \quad 1+2+30 = 33 \text{ słów,}$$

$$DV2 \quad 1+5*(2 + 2 + 6) = 51 \text{ słów,}$$

$$A45 \quad 1 \text{ słowo,}$$

$$YQ \quad 16 * 2 = 32 \text{ słowa.}$$

Wobec tego długość pola dokumentu wynosi 51 słów. Pozycja MAX dokumentu DV2 rozmieszczona jest na tych samych miejscach co pozycja MA. Pozycja MAX zajmuje $1 + 3 * (2 + 2) + 11 = 24$ słowa, a pozycja MA $5*(2 + 2 + 6) = 50$ słów.

Dokument DV1 zawiera znacznik równy 4 (pozycja Q). Cały dokument A45 stanowi znacznik o wartości 7. Dokumenty DV2 oraz XQ nie mają znacznika i zawierają jednakową liczbę pozycji.

Zbiór XPRS jest zbiorem sortowania. Zbiór ten ma tylko jeden dokument o nazwie WBT. Długość

pola dokumentu tego zbioru wynosi $4 * 7 = 28$ słów, bowiem zbiorowi temu w części ENVIRON-
MENT przydzielono 4 taśmy sortowania.

Sekcja roboczych składa się

- z macierzy ROB1 - 12 słów,
- ze stałej ST o wartości 1.3,
- z dokumentów DOK oraz POL.

Każdy z tych dokumentów ma oddzielne pole o długości:

DOK 9 słów,

POL 5 słów.

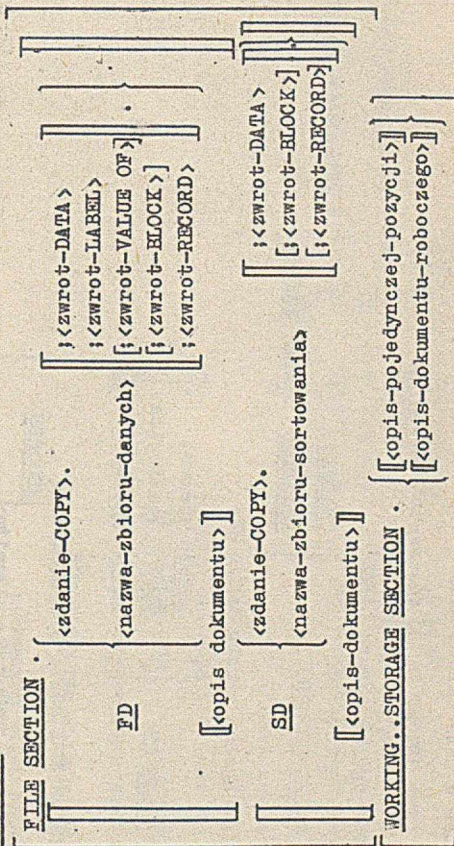
Pozycje dokumentu DOX rozmieszczone są w tym samym polu co dokument DOK.

Dokument DOX zajmuje też 9 słów.

W K Ł A D K A II

1. Postać ogólna części - Dane

DATA DIVISION .



1.1. Postać ogólna zwrotu DATA

DATA RECORD IS [, <nazwa-dokumentu>]

1.2. Postać ogólna zwrotu LABEL

LABEL RECORD IS { OMITTED
STANDARD [WITH GENER] }

1.3. Postać ogólna zwrotu VALUE OF

VALUE OF [, IDENT <tekst>
SECT <całkowita>
SECUR <całkowita>]

1.4. Postać ogólna zwrotu BLOCK

BLOCK CONTAINS <całkowita> { RECORDS
CHARACTERS }

1.5. Postać ogólna zwrotu RECORD

RECORD CONTAINS <całkowita> CHARACTERS

1.6. Postać ogólna zdania COPY

COPY <nazwa-biblioteczna>

1.7. Postać ogólna opisu dokumentu

I. Dokument będący pozycją elementarną

$$\left\{ \begin{matrix} 1 \\ 01 \end{matrix} \right\} \langle \text{nazwa-dokumentu} \rangle . \langle \text{opis-pozycji} \rangle$$

II. Dokument zawierający inne pozycje

 $\begin{Bmatrix} 1 \\ 01 \end{Bmatrix} \langle \text{nazwa-dokumentu} \rangle.$
$$\begin{aligned} & \left\{ \begin{array}{l} \langle \text{nazwa-pozycji} \rangle \\ \text{FILLER} \end{array} \right\} \left[\left[\left[\left[\langle \text{nr-poziomu-grupowej} \rangle ; \right. \right. \right. \\ & \left. \left. \left. \left[\langle \text{nr-poziomu-elementarnej} \rangle ; \right. \right. \right. \right. \\ & \left. \left. \left. \left. \left. \left[\langle \text{zwrot-OCCURS} \rangle \right] \right] \right] \right] \left[\langle \text{zwrot-REDEFINES} \rangle \right] \right] \right] \left[\langle \text{zwrot-OCCURS} \rangle \right] \right] \\ & \left\{ \begin{array}{l} \langle \text{nazwa-pozycji} \rangle \\ \text{FILLER} \end{array} \right\} \left[\left[\left[\left[\langle \text{nr-poziomu-grupowej} \rangle ; \right. \right. \right. \\ & \left. \left. \left. \left[\langle \text{nr-poziomu-elementarnej} \rangle ; \right. \right. \right. \right. \\ & \left. \left. \left. \left. \left. \left[\langle \text{zwrot-OCCURS} \rangle \right] \right] \right] \right] \left[\langle \text{zwrot-REDEFINES} \rangle \right] \right] \right] \left[\langle \text{opis-pozycji} \rangle \right] \right] \\ & \left. \left. \left. \left. \left. \left[\langle \text{deklaracja-nazwy-warunku} \rangle \right] \right] \right] \right] \right] \right] \end{aligned}$$

1/ Pozycja opisana zwrotem REDEFINES nie może zawierać pozycji opisanej tym zwrotem.

III. Opis dokumentu pobierany z biblioteki

$\left\{ \begin{matrix} 1 \\ 01 \end{matrix} \right\} \langle \text{nazwa-dokumentu} \rangle . \langle \text{zdanie-COPY} \rangle .$

1.8. Postać ogólna opisu dokumentu roboczego

I. Dokument będący pozycją elementarną

$\left\{ \begin{matrix} 1 \\ 01 \end{matrix} \right\} \langle \text{nazwa-dokumentu} \rangle [; \langle \text{zwrot-REDEFINES} \rangle^{1/}] \langle \text{opis-pozycji} \rangle$
 $[[\langle \text{deklaracja-nazwy-warunku} \rangle]]$

II. Dokument zawierający inne pozycje

$\left\{ \begin{matrix} 1 \\ 01 \end{matrix} \right\} \langle \text{nazwa-dokumentu} \rangle [; \langle \text{zwrot-REDEFINES} \rangle^{1/}]$
 $[[[[\langle \text{nr-poziomu-grupowej} \rangle ;$
 $\left\{ \begin{matrix} \langle \text{nazwa-pozycji} \rangle \\ \text{FILLER} \end{matrix} \right\} [; \langle \text{zwrot-REDEFINES} \rangle^{1/}] [; \langle \text{zwrot-OCCURS} \rangle .]]$
 $[[[\langle \text{nr-poziomu-elementarnej} \rangle ;$
 $\left\{ \begin{matrix} \langle \text{nazwa-pozycji} \rangle \\ \text{FILLER} \end{matrix} \right\} [; \langle \text{zwrot-REDEFINES} \rangle^{1/}] \langle \text{opis-pozycji} \rangle$
 $[; \langle \text{zwrot-OCCURS} \rangle .]] [\langle \text{deklaracja-nazwy-warunku} \rangle]]]]$

1/ Pozycja opisana zwrotem REDEFINES nie może zawierać . pozycji opisanej tym zwrotem.

III. Opis dokumentu pobierany | z biblioteki

$\left\{ \begin{matrix} 1 \\ 01 \end{matrix} \right\} \langle \text{nazwa-dokumentu} \rangle . \langle \text{zdanie-COPY} \rangle .$

1.9. Postać ogólna deklaracji nazwy warunku

88 $\langle \text{nazwa-warunku} \rangle ; \text{VALUE IS } \left[\begin{matrix} 1 \\ 1 \end{matrix} \right] \langle \text{stała} \rangle \left[\text{THRU } \langle \text{stała} \rangle \right] \left[\begin{matrix} 1 \\ 1 \end{matrix} \right] .$

1.10. Postać ogólna opisu pojedynczej pozycji

77 $\langle \text{nazwa-pozycji} \rangle \left[; \langle \text{zwrot-REDEFINES} \rangle \right] \langle \text{opis-pozycji} \rangle \left[; \langle \text{zwrot-OCCURS} \rangle \right] .$

1.11. Postać ogólna opisu pozycji

$\left\{ \begin{matrix} \langle \text{zwrot-FIXED} \rangle \\ \langle \text{zwrot-FIXED2} \rangle \\ \langle \text{zwrot-FLOAT} \rangle \\ \langle \text{zwrot-PICTURE} \rangle \left[; \langle \text{zwrot-BLANK} \rangle \right] \left[; \langle \text{zwrot-JUSTIFIED} \rangle \right] \end{matrix} \right\} \left[; \langle \text{zwrot-VALUE} \rangle \right] .$

1/ Zwrot VALUE jest ignorowany jeżeli pozycja została opisana zwrotem OCCURS.

1.12. Postać ogólna zwrotu FIXED

FIXED

1.13. Postać ogólna zwrotu FIXED2

FIXED2

1.14. Postać ogólna zwrotu FLOAT

FLOAT

1.15. Postać ogólna zwrotu PICTURE

$\left\{ \begin{array}{c} \text{PICTURE} \\ \text{PIC} \end{array} \right\}$ IS <obraz>

1.16. Postać ogólna zwrotu BLANK WHEN ZERO

BLANK WHEN ZERO

1.17. Postać ogólna zwrotu JUSTIFIED RIGHT

JUSTIFIED RIGHT

1.18. Postać ogólna zwrotu VALUE

VALUE IS <stała>

1.19. Postać ogólna zwrotu OCCURS

OCCURS <całkowita> TIMES

1.20. Postać ogólna zwrotu REDEFINES

REDEFINES <nazwa-danej>

4.4. PROCEDURE DIVISION - PROCEDURY

4.4.1. STRUKTURA CZĘŚCI PROCEDURALNEJ

PROCEDURE DIVISION składa się z tzw. części DECLARATIVES i właściwej części proceduralnej.

Część DECLARATIVES zawiera opis niestandardowych czynności, które mają być wykonane przy opracowywaniu metryk zbiorów lub w razie wykrycia błędu przy pisaniu bądź czytaniu danych (zobacz opis zdania USE). Składa się z jednej lub więcej sekcji poprzedzonych słowem kluczowym DECLARATIVES i kończy się zdaniem END DECLARATIVES.

Właściwa część proceduralna określa czynności, które mają być wykonane na danych opisanych w DATA DIVISION, przy wykorzystaniu urządzeń przydzielonych w ENVIRONMENT DIVISION. Część proceduralna kończy się zdaniem END PROGRAM.

Postać ogólna części proceduralnej podana jest we wkładce na końcu rozdziału. (Wkładka III).

4.4.2. INSTRUKCJE I ZDANIA, PARAGRAFY I SEKCJE

Jak widać z postaci ogólnej części proceduralnej składa się ona ze zdań. Zdaniem może być instrukcja zakończona kropką, ciąg instrukcji oddzielonych średnikami i zakończony kropką lub zdanie ENTER, zdanie EXIT, zdanie USE, zdanie COPY i zdanie END.

Instrukcja jest to syntaktycznie poprawna kombinacja słów i ograniczników zaczynająca się od słowa kluczowego języka COBOL.

Zdania mogą być grupowane w paragrafy. Paragraf zaczyna się nazwą paragrafu, po której występuje kropka, a zakończony jest nazwą następnego paragrafu lub nagłówkiem sekcji, lub zdaniem END.

Postać:

<nazwa-paragrafu>.

Jeżeli w części proceduralnej występują paragrafy, to żadne zdanie nie może występować poza paragrafem. Zdania bądź paragrafy mogą tworzyć sekcje. Sekcja zaczyna się od nagłówka sekcji, który ma następującą postać:

<nazwa-sekcji> SECTION [<numer-segmentu>].

a kończy się na najbliższym nagłówku sekcji lub zdaniem END.

<numer-segmentu> związany jest z segmentacją części proceduralnej. Jeśli w części proceduralnej występują sekcje, to żaden paragraf ani tym bardziej zdanie, nie może wystąpić poza sekcją.

Ponieważ część DECLARATIVES składa się z sekcji, to wystąpienie jej powoduje, że cała część proceduralna musi składać się z sekcji.

Sekcje i paragrafy nazywają się procedurami, a ich nazwy - nazwami proceduralnymi.

Jak wiadomo, w poprzednich częściach programu COBOL nazwy sekcji są słowami zastrzeżonymi i programiście nie wolno ich zmieniać. W części proceduralnej nazwy sekcji i paragrafów są dowolne.

4.4.3. REGUŁY PRZESTANKOWANIA

Każde zdanie w części proceduralnej musi kończyć się kropką. Jeżeli zdanie składa się z kilku instrukcji, to kolejne instrukcje oddziela się od siebie średnikami. Nie należy jednak budować zdań zbyt długich, gdyż to może utrudniać translację programu.¹⁾ Jeżeli w zdaniu występuje lista argumentów, to oddziela się je od siebie przecinkami.

Dokładne formaty poszczególnych instrukcji podane są w punkcie 'Opis instrukcji' oraz we wkladce na końcu tego rozdziału. (Wkładka III). Alfabetyczny spis instrukcji Języka COBOL dla maszyny ZAM 41 podany jest w Dodatku B.

4.4.4. SEGMENTACJA PROGRAMU

W języku COBOL segmentować można tylko część proceduralną. Jeżeli program wynikowy wraz ze stałymi, roboczymi i polami zbiorów mieści się w pamięci operacyjnej maszyny, to nie ma sensu dzielenie go na segmenty. W przeciwnym przypadku podział na segmenty jest konieczny. Tworzy się wtedy jeden segment stały, będący w pamięci operacyjnej przez cały czas wykonywania programu oraz pewną liczbę segmentów wymiennych,

-
- 1) Długość zdania w części proceduralnej ograniczona jest długością bufora wejściowego translatora, w którym zdanie powinno mieścić się w całości. W przeciwnym przypadku sygnalizowany jest błąd, który można usunąć dzieląc zdanie na dwa lub więcej. Jeżeli program pisany jest na maszynę ZAM 41 z dużą pamięcią (15K) - ograniczenie to nie obowiązuje.

które są sprowadzane do pamięci operacyjnej na czas wykonania. Sprowadzenie do pamięci segmentu wymiennego powoduje wymazanie umieszczonego w niej poprzednio innego segmentu wymiennego. Ważne jest, że segment wymienny sprowadzany jest do pamięci operacyjnej w swojej pierwotnej postaci i z tego faktu wynikają ograniczenia wykonywania w segmentach wymiennych instrukcji `ALTER` i `PERFORM`. Przejście do wykonywania segmentu wymiennego, którego aktualnie nie ma w pamięci operacyjnej, poprzedzone jest sprowadzeniem tego segmentu do pamięci. Podział części proceduralnej odbywa się przez nadanie sekcjom odpowiednich numerów. Jak pamiętamy, w nagłówku sekcji znajduje się opcjonalny <numer-segmentu>, który jest liczbą całkowitą bez znaku, przyjmującą wartości od 0 do 99. Wszystkie sekcje oznaczone numerami od 0 do 49 tworzą segment stały. Jeżeli brak numeru segmentu w nagłówku sekcji, to przyjmuje się, że jest on równy 0. Sekcje oznaczone tym samym numerem n większym od 49, a mniejszym od 100 tworzą segment wymienny o numerze n . Ponieważ podstawą podziału na segmenty są numery umieszczone w nagłówkach sekcji, segmentować można tylko programy, których część proceduralna podzielona jest na sekcje.

Przy podziale na segmenty ważne jest, aby te sekcje, które są wykonywane często lub są potrzebne podczas całego wykonywania programu były umieszczone w segmencie stałym. Sekcje tworzące segment wymienny powinny w miarę możliwości stanowić logiczną całość. Sekcje wchodzące w skład segmentu wymiennego nie muszą występować

w programie obok siebie. Część DECLARATIVES segmentuje się w ten sam sposób co właściwą część proceduralną z tym, że sekcje jej jako wykonywane w programie rzadko - są zazwyczaj umieszczane w segmentach wymiennych.

Przy dołączaniu w segmencie wymiennym symbolicznych wejść i wyjść (za pomocą języka wstawek) należy pamiętać, że można z nich korzystać tylko w tym segmencie wymiennym, w którym zostały dołączone i że po każdorazowym wprowadzeniu do pamięci operacyjnej tego segmentu buforów wejść i wyjść będą puste. Wymazanie segmentu wymiennego przez sprowadzenie na jego miejsce innego segmentu wymiennego powoduje zniszczenie zawartości buforów. Z symbolicznych wejść i wyjść dołączonych w segmencie stałym można korzystać w całym programie, tak jak i z symbolicznego wejścia i wyjścia o numerze 0 (zero), które dołączane są automatycznie poza programem.

4.4.5. ZDANIE WARUNKOWE, BEZWARUNKOWE I STERUJĄCE TRANSLACJĄ

Wykonanie programu zaczyna się od pierwszej instrukcji w PROCEDURE DIVISION nie należącej do części DECLARATIVES. Pierwsza instrukcja wykonywana może należeć do segmentu wymiennego. Wykonanie programu kończy się po napotkaniu instrukcji STOP RUN. Instrukcje wykonywane są w takiej kolejności w jakiej zostały napisane. Kolejność wykonania zmieniana jest przez instrukcje sterujące, przekazujące sterowanie do określonej procedury, jak np. instrukcja GO TO. Jeżeli instrukcja powoduje sprawdzenie czy spełniony jest wymieniony w niej warunek i zależnie od

tego przekazuje sterowanie programu w różne miejsca - wtedy instrukcja taka jest instrukcją warunkową..

W języku COBOL są następujące instrukcje warunkowe:

instrukcja IF

instrukcja READ

instrukcja RETURN

instrukcja COMPUTE z frazą SIZE ERROR

instrukcja PERFORM z frazą UNTIL lub TIMES.

Wszystkie pozostałe instrukcje (w tym instrukcja COMPUTE bez frazy SIZE ERROR i instrukcja PERFORM bez frazy UNTIL lub TIMES) są instrukcjami bezwarunkowymi. Instrukcje bezwarunkowe powodują wykonanie w programie określonej czynności (bez sprawdzania warunku).

Instrukcja warunkowa zakończona kropką tworzy zdanie warunkowe. W zdaniu warunkowym przed instrukcją warunkową może wystąpić ciąg instrukcji bezwarunkowych. Jeżeli zdanie składa się z samych instrukcji bezwarunkowych lub jednej instrukcji bezwarunkowej zakończonej kropką, to jest to zdanie bezwarunkowe. Oczywiście instrukcje tworzące zdanie warunkowe czy bezwarunkowe muszą być od siebie oddzielone średnikami.

Prócz zdań bezwarunkowych i warunkowych w programie COBOL mogą wystąpić zdania sterujące translacją. Są to:

zdanie ENTER

zdanie NOTE

zdanie EXIT

zdanie COPY

zdanie END

Każde ze zdań musi być w programie zakończone kropką.

4.4.6. WYRAŻENIA ARYTMETYCZNE I WARUNKI. KWA- LIFIKOWANIE I INDEKSOWANIE NAZW

4.4.6.1. Wyrażenia arytmetyczne

Wyrażeniem arytmetycznym jest nazwa elementarnej jednostki danych typu FIXED, FIXED2 lub FLOAT, lub liczba bez znaku. Prócz tego w wyrażeniu arytmetycznym mogą wystąpić symbole działań arytmetycznych i nawiasy.

W języku COBOL przyjęto następujące symbole działań arytmetycznych:

działanie	symbol
dodawanie	+
odejmowanie	-
mnożenie	*
dzielenie	/
potęgowanie	**

Wszystkie powyższe działania wykonywane są na dwu operandach. Jedynie dodawanie i odejmowanie mogą się odnosić do jednego operandu np. +77, -A i wtedy nazywane są dodawaniem prefiksowym i odejmowaniem prefiksowym. Działania dodawania i odejmowania na dwóch operandach noszą nazwę dodawania infiksowego i odejmowania infiksowego. Powyższe rozróżnienie potrzebne jest do ustalenia listy mocy działań arytmetycznych, która przedstawia się następująco:

- 1) dodawanie prefiksowe i odejmowanie prefiksowe,
- 2) potęgowanie,
- 3) mnożenie, dzielenie,
- 4) dodawanie infiksowe, odejmowanie infiksowe.

Działaniem większej mocy jest działanie stojące na liście pod niższym numerem.

Przy wyliczaniu wartości wyrażenia arytmetycznego wykonuje się najpierw działanie zamknięte w największej liczbie nawiasów, przy równej liczbie nawiasów działanie o większej mocy, a w przypadku równej mocy i równej liczby nawiasów wykonuje się działania od lewej strony - kolejno.

Działania arytmetyczne mogą być wykonywane tylko na operandach typu FIXED, FIXED2 i FLOAT, przy czym typ FLOAT jest mocniejszy od typu FIXED2 i FIXED, a FIXED2 od FIXED. Znaczy to, że jeśli działanie arytmetyczne wykonywane jest na dwóch operandach różnego typu, to operand typu słabszego jest przekształcany do postaci typu mocniejszego i wynik działania jest też typu mocniejszego. Jeśli obydwa operandy są jednego typu, to wynik jest też tego typu. Jedynym wyjątkiem jest potęgowanie, w którym podstawa może być typu dowolnego ale wykładnik musi być typu FIXED. Typ wyniku jest taki, jak typ podstawy.

Uwaga

W wyrażeniach, jako operandy, mogą wystąpić następujące nazwy zastrzeżone: stała figuratywna

ZERO, nazwy specjalnych rejestrów TALLY i DRUM oraz nazwy pozycji metrykowych SECT, CREAT, EXPIR, GENER gdyż są one wielkościami typu FIXED. Nazwy pozycji metrykowych mogą jednak występować tylko w wyrażeniach wykonywanych w ramach niestandardowych procedur metrykowych w części DECLARATIVES.

4.4.6.2. Warunki

Warunkiem jest porównanie wielkości, sprawdzenie klasy, nazwa warunku oraz nazwa wskaźnika. W warunku mogą wystąpić nawiasy oraz działania logiczne.

4.4.6.2.1. Porównanie wielkości

Porównanie wielkości polega na sprawdzeniu czy między dwoma operandami zachodzi określona relacja, co daje w wyniku odpowiedź "prawda" lub "fałsz" w przypadku przeciwnym. Porównanie wielkości ma następującą postać ogólną:

$$\left\{ \begin{array}{l} \langle \text{nazwa-danej-1} \rangle \\ \langle \text{tekst-1} \rangle \\ \langle \text{wyrażenie-arytmetyczne} \rangle \end{array} \right\} \text{ IS } [\text{NOT}] \left\{ \begin{array}{l} \text{GREATER} \\ \text{LESS} \\ \text{EQUAL} \end{array} \right\} \text{ THAN } \left\{ \begin{array}{l} \langle \text{nazwa-danej-2} \rangle \\ \langle \text{tekst-2} \rangle \\ \langle \text{wyrażenie-arytmetyczne} \rangle \end{array} \right\}$$

Znaczenie słów kluczowych określających relację jest następujące:

	słowo kluczowe	relacja
IS	[NOT] <u>GREATER</u> THAN	[nie] większe niż
IS	[NOT] <u>LESS</u> THAN	[nie] mniejsze niż
IS	[NOT] <u>EQUAL</u>	[nie] równe

Jak widać z postaci ogólnej operandy porównania wielkości mogą być nazwami danych, tekstami lub wyrażeniami arytmetycznymi. Operandy mogą być w postaci znakowej lub obliczeniowej. Nazwy elementarnych jednostek danych opisanych zwrotem PICTURE, teksty oraz stałe figuratywne HIGH..VALUE, LOW..VALUE i SPACE są operandami znakowymi. Operandami obliczeniowymi są wyrażenia arytmetyczne. Obydwa operandy porównania muszą być albo znakowe albo obliczeniowe.

Porównanie operandów znakowych

Kolejne znaki obydwu operandów od lewej do prawej porównywane są parami, zgodnie z uporządkowaniem znaków kodu KW8 (podanym w Dodatku C). Gdy operandy są jednakowej długości i wszystkie pary znaków są jednakowe - operandy są równe. Wystąpienie pierwszej pary znaków nierównych określa relację nierówności między operandami. Większym operandem jest ten, który zawiera znak o wyższej wartości w kodzie KW8. Gdy operandy są różnej długości porównanie przebiega tak samo, aż do wyczerpania znaków operandu krótszego. Jeśli do tego momentu operandy były równe tj. nie wystąpiły znaki różne, określające relację między operandami, to jeżeli pozostałe znaki operandu dłuższego są różne od spacji - operand dłuższy jest większy. Gdy pozostałe znaki operandu dłuższego są spacjami - oba operandy są równe.

Porównanie operandów obliczeniowych

Porównywana jest algebraiczna wartość operandów, przy czym gdyby operandy były różnego typu - operand typu słabszego jest przed porównaniem zamieniany na postać typu mocniejszego.

4.4.6.2.2. Sprawdzenie klasy

Sprawdzenie klasy* polega na zbadaniu czy aktualna wartość znakowej jednostki danych, tj. opisanej zwrotem PICTURE, jest klasy numerycznej (składa się tylko z cyfr ewentualnie poprzedzonych znakiem + lub -) czy klasy alfabetycznej (składa się tylko z liter lub spacji).

Sprawdzenie klasy ma następującą postać ogólną:

$$\langle \text{nazwa-danej} \rangle \text{ IS } [\text{NOT}] \left\{ \begin{array}{l} \text{NUMERIC} \\ \text{ALPHABETIC} \end{array} \right\}$$

Wynikiem sprawdzenia klasy jest "prawda" gdy jednostka danych [nie] jest klasy wymienionej i "fałsz" w przypadku przeciwnym.

Badana jednostka danych musi być elementarna i znakowa. Nie wolno sprawdzać czy są klasy ALPHABETIC - jednostki danych opisane obrazem liczbowym lub liczbowym wydawniczym.

4.4.6.2.3. Nazwa warunku

Zmienna <nazwa-warunku> przyjmuje wartość "prawda", gdy zmienna, której nazwa warunku została przypisana - przyjmuje wartość ze zbioru wartości, określonego dla tej nazwy warunku. W prze-

ciwnym przypadku nazwa warunku przyjmuje wartość "fałsz".

Postać ogólna:

<nazwa-warunku>

4.4.6.2.4. Nazwa wskaźnika

Zmienna <nazwa-wskaźnika> przyjmuje wartość "prawda" lub "fałsz" w zależności od wartości odpowiadającego jej wskaźnika (bitu trzeciego słowa maszyny) oraz od tego czy została opisana słowem ON czy OFF (por. pkt. 4.2.1.1.).

Postać ogólna:

<nazwa-wskaźnika>

4.4.6.2.5. Działania logiczne

W warunku mogą wystąpić trzy działania logiczne, określone następującymi słowami kluczowymi:

słowo kluczowe	działanie
<u>NOT</u>	negacja
<u>AND</u>	iloczyn logiczny
<u>OR</u>	suma logiczna

Jeśli W1 i W2 są warunkami, to wynik wykonania działań logicznych na tych warunkach jest następujący:

Warunek		W a r t o ś ć		
W1	W2	NOT W1	W1 AND W2	W1 OR W2
prawda	prawda	fałsz	prawda	prawda
fałsz	prawda	prawda	fałsz	prawda
prawda	fałsz	fałsz	fałsz	prawda
fałsz	fałsz	prawda	fałsz	fałsz

4.4.6.2.6. Moc i kolejność wykonywania działań występujących w warunkach

Przyjmuje się następującą listę mocy działań występujących w warunkach:

- 1-5) działania arytmetyczne,
- 6) relacje, ALPHABETIC, NUMERIC,
- 7) negacja,
- 8) iloczyn logiczny,
- 9) suma logiczna.

Działaniem większej mocy będziemy nazywali działanie stojące na powyższej liście pod mniejszym numerem. Moc działań arytmetycznych opisana jest w punkcie 'Wyrażenia arytmetyczne'.

Z dwu sąsiednich działań wykonuje się najpierw to, które jest zamknięte w większej liczbie nawiasów; w przypadku równej liczby nawiasów najpierw wykonuje się działanie o większej mocy; w przypadku równej mocy i jednakowej liczby nawiasów najpierw wykonuje się działanie występujące w warunku z lewej strony drugiego.

4.4.6.3. Kwalifikowanie nazw

Każda nazwa danej, warunku czy procedury w części proceduralnej programu musi dotyczyć tylko

jednego obiektu. Osiąga się to przez nadanie różnych nazw różnym obiektom lub przez wyliczenie nazw wyższych elementów hierarchii, w której występuje dany obiekt. W tym przypadku wyższe elementy hierarchii stają się kwalifikatorami nazwy, a proces ujednoznacznienia nazywamy kwalifikacją. Nazwa kwalifikowana ma następującą postać:

$$\langle \text{identyfikator-1} \rangle \left[\left\{ \begin{array}{c} \underline{\text{IN}} \\ \underline{\text{OF}} \end{array} \right\} \langle \text{identyfikator-k} \rangle \right]$$

Kolejne kwalifikatory są to coraz wyższe elementy hierarchii zawierającej nazwę kwalifikowaną czyli $\langle \text{identyfikator-1} \rangle$. Oczywiście przy kwalifikacji nie musimy wymieniać wszystkich kolejnych poziomów hierarchii a tylko tyle, ile trzeba, aby nazwa stała się jednoznaczna. Nazwy zbiorów danych i zbiorów sortowania są najwyższymi kwalifikatorami nazw danych i warunków; wobec tego same nie mogą już być kwalifikowane i muszą być w programie jednoznaczne. To samo odnosi się do nazw sekcji, które są najwyższymi kwalifikatorami nazw proceduralnych i muszą być w programie jednoznaczne.

Wszystkie nazwy kwalifikowane ujednoznaczniające nazwę są równoważne.

Nazwa danej użyta jako kwalifikator nie może być nazwą indeksowaną.

Przykład

Założmy, że w DATA DIVISION opisano następujący dokument:

01 KARTA..PRACY.

02 PRACOWNIK.

03 NAZWISKO; PIC 'A(20)'.

03 IMIE; PIC 'A(15)'.

02 OJCIEC

03 IMIE; PIC 'A(15)'.

02 MATKA

03 NAZWISKO..PANIENSKIE; PIC 'A(20)'.

03 IMIE; PIC 'A(15)'.

Ponieważ nazwa pozycji elementarnej tego dokumentu "IMIE" jest niejednoznaczna, musi być w programie kwalifikowana. Tak więc nazwa IMIE IN PRACOWNIK dotyczy pozycji zawierającej imię właściciela karty, IMIE OF OJCIEC dotyczy pozycji zawierającej imię ojca właściciela karty, a IMIE IN MATKA - pozycji zawierającej imię matki. Można by użyć innych jeszcze nazw kwalifikowanych np. IMIE IN MATKA IN KARTA..PRACY, która jest równoznaczna z nazwą IMIE OF MATKA.

4.4.6.4. Indeksowanie nazw

Indeksy występują w odwołaniach do konkretnych składowych macierzy.

Indeksowana nazwa danej ma następującą postać ogólną:

$$\langle \text{identyfikator-1} \rangle \left\{ \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right\} \langle \text{identyfikator-k} \rangle \left(\left[\langle \text{indeks} \rangle \right] \right)$$

Indeksem może być dowolne wyrażenie arytmetyczne, którego wartość jest całkowita i większa od zera. Mogą więc być indeksami liczby całkowite

oraz nazwy wielkości typu FIXED: np. nazwa rejestru specjalnego TALLY lub nazwa jednostki danych opisana zwrotem FIXED.

Nazwa jednostki danych będąca indeksem może być kwalifikowana lub indeksowana.

Liczba wymiarów macierzy równa jest liczbie zwrotów OCCURS występujących w opisie tej macierzy i w opisie pozycji zawierających tę macierz. Na przykład następujący opis:

03 X; OCCURS 2.

04 Y; OCCURS 3.

05 Z; FLOAT; OCCURS 3.

tworzy taką strukturę:

$$\begin{array}{l}
 \mathbf{X}(1) \left\{ \begin{array}{l}
 \mathbf{Y}(1,1) \left\{ \begin{array}{l} \mathbf{Z}(1,1,1) \\ \mathbf{Z}(1,1,2) \\ \mathbf{Z}(1,1,3) \end{array} \right. \\
 \mathbf{Y}(1,2) \left\{ \begin{array}{l} \mathbf{Z}(1,2,1) \\ \mathbf{Z}(1,2,2) \\ \mathbf{Z}(1,2,3) \end{array} \right. \\
 \mathbf{Y}(1,3) \left\{ \begin{array}{l} \mathbf{Z}(1,3,1) \\ \mathbf{Z}(1,3,2) \\ \mathbf{Z}(1,3,3) \end{array} \right.
 \end{array} \right. \\
 \\
 \mathbf{X}(2) \left\{ \begin{array}{l}
 \mathbf{Y}(2,1) \left\{ \begin{array}{l} \mathbf{Z}(2,1,1) \\ \mathbf{Z}(2,1,2) \\ \mathbf{Z}(2,1,3) \end{array} \right. \\
 \mathbf{Y}(2,2) \left\{ \begin{array}{l} \mathbf{Z}(2,2,1) \\ \mathbf{Z}(2,2,2) \\ \mathbf{Z}(2,2,3) \end{array} \right. \\
 \mathbf{Y}(2,3) \left\{ \begin{array}{l} \mathbf{Z}(2,3,1) \\ \mathbf{Z}(2,3,2) \\ \mathbf{Z}(2,3,3) \end{array} \right.
 \end{array} \right.
 \end{array}$$

Na powyższym rysunku widać wyraźnie, że pozycja X jest macierzą jednowymiarową o 2 składowych, pozycja Y macierzą dwuwymiarową o 6 składowych, a pozycja Z macierzą trójwymiarową o 18 składowych, z których każda jest pozycją elementarną typu FLOAT.

Pierwszy indeks odpowiada najbardziej zewnętrznemu zwrotowi OCCURS, drugi indeks - drugiemu z kolei zwrotowi OCCURS itd. a ostatni indeks najbardziej wewnętrznemu zwrotowi OCCURS odnoszącemu się do tej zmiennej. Liczba wymiarów macierzy nie jest ograniczona (porównaj opis zwrotu OCCURS).

W odwołaniu do określonej składowej macierzy musi występować dokładnie tyle indeksów ile zwrotów OCCURS opisuje daną macierz. Najmniejszą wartością indeksu jest 1, a największą jest <całkowita> wymieniona w zwrocie OCCURS, z którym związany jest dany indeks. Przekroczenie tego zakresu przez którykolwiek indeks zmiennej indeksowanej powoduje, że wynik takiego odwołania do zmiennej indeksowanej jest nieokreślony, ale błąd nie jest sygnalizowany.

Przy indeksowaniu nazwy kwalifikowanej indeksy pisze się po ostatnim kwalifikatorze, np.:

A IN B IN C (1, 2, I + 1)

Powyższe odwołanie dotyczy elementu trójwymiarowej macierzy A, kwalifikowanej nazwami B i C.

Parametrem formalnym jest każdy identyfikator występujący i niezadeklarowany we wstawce.

W powyższym przykładzie parametrem formalnym nie jest identyfikator Y, który jest we wstawce zadeklarowany Y) +103, zaś identyfikator X jest parametrem formalnym i wystąpił w zwrocie REPLACING. Każdy parametr formalny użyty we wstawce musi wystąpić w zwrocie REPLACING, w którym musi mieć przyporządkowaną nazwę danej. W czasie tłumaczenia wszystkie parametry formalne zastępowane są adresami odpowiadających im nazw danych. Wstawka kończy się zwrotem ENTER COBOL, za którym następuje kropka. Zwrot ENTER COBOL musi być napisany w osobnym wierszu.

4.4.7.2. Zdanie NOTE

Zdanie NOTE umożliwia programiście pisanie komentarzy wewnątrz programu w języku COBOL.

Zdania NOTE wypisywane są na tabulogramach programu (porównaj opis dyrektywy TAB w rozdziale pt. "Język operacyjny"), ale nie są włączane do programu wynikowego. Zdanie NOTE ma następującą postać ogólną:

NOTE <komentarz>

Przykład

NOTE KONIEC CZYTANIA DOKUMENTÓW ZBIORU
'DANE..WEJSCIOWE'.

Za komentarz uważany jest ciąg znaków w kodzie KW6 występujących między słowem NOTE a kropką. Z tego powodu wewnątrz komentarza nie może wys-

tać pojedyncza kropka. Można jednak użyć w komentarzu łącznika '...' (jak w przykładzie 'DANE..WEJSCIOWE').

4.4.7.3. Zdanie EXIT

Zdanie EXIT służy do określenia wspólnego punktu wyjścia dla ciągu procedur, do których przekazano sterowanie w wyniku wykonania instrukcji PERFORM.

Postać ogólna

<nagłówek-paragrafu> EXIT

Zdanie EXIT musi być jedynym zdaniem paragrafu.

Przykład

```

:
PERFORM A THRU B.
:
A. :
:
IF X LESS 2 THEN GO TO B ELSE NEXT.
C. :
:
IF X GREATER 0 THEN GO TO B ELSE NEXT.
:
B. EXIT.
```

Sekcja w części DECLARATIVES lub ciąg procedur, których wykonanie zostało wywołane przez instrukcję PERFORM - tworzą zakres tej instrukcji PERFORM (w przykładzie są to paragrafy A, C i B). Wykonanie zakresu instrukcji PERFORM

kończy się po wykonaniu procedury określonej w instrukcji jako koniec zakresu (w przykładzie jest to paragraf B), po czym następuje powrót do instrukcji następnej po instrukcji PERFORM - czyli do punktu wywołania.

Jeżeli w zakresie instrukcji PERFORM występują instrukcje, które mają przekazać sterowanie do punktu wywołania (w przykładzie instrukcji IF) - to trzeba je zapisać jako instrukcje przekazyjące sterowanie do zdania EXIT kończącego ten zakres. W innych przypadkach zdanie EXIT ma znaczenie tylko komentarzowe.

4.4.7.4. Zdanie USE

Zdanie USE określa niestandardową procedurę metrykową lub niestandardową procedurę błędu, wykonywaną dodatkowo obok odpowiedniej procedury standardowej w systemie wejścia-wyjścia. Zdanie USE ma dwie wersje: I dotyczy niestandardowej procedury błędu, II niestandardowej procedury metrykowej.

Postać ogólna obu wersji zdania USE podana jest we wkładce III na końcu tego rozdziału.

Przykłady

USE AFTER STANDARD ERROR PROCEDURE ON
EWIDENCJA.

USE BEFORE BEGINNING FILE LABEL PROCEDURE ON
INPUT.

Zdanie USE - jeżeli występuje w programie - musi być umieszczone w części DECLARATIVES, bezpośrednio po nagłówku sekcji. Zdania lub paragrafy wchodzące w skład tej sekcji stanowią treść niestandardowej procedury określonej w zdaniu USE. Każde zdanie USE musi być napisane w osobnej sekcji w części DECLARATIVES. Zdanie USE nie jest wykonywane natomiast określa warunki w jakich będzie wykonywana nie - standardowa procedura. Jeżeli w zdaniu USE występuje słowo INPUT, to niestandardowa procedura będzie wykonywana dla wszystkich zbiorów wejściowych, a jeżeli słowo OUTPUT - to odpowiednio dla wszystkich zbiorów wyjściowych. Jeżeli w zdaniu USE występuje nazwa zbioru, to niestandardowa procedura wykonywana jest tylko dla tego zbioru. Jeżeli do pewnego zbioru odnosi się zdanie USE, w którym nazwa zbioru występuje jawnie i zdanie USE (w tej samej wersji), w którym zbiór ten jest określony słowem INPUT lub OUTPUT, to dla tego zbioru będzie wykonywana tylko niestandardowa procedura określona zdaniem USE, w którym nazwa zbioru wystąpiła jawnie. Nazwa zbioru występująca w zdaniu USE nie może być nazwą zbioru sortowania.

W sekcji USE wolno się odwoływać do procedur znajdujących się we właściwej części proceduralnej z tym, że sterowanie musi wrócić do sekcji USE, gdyż inaczej wynik działania programu będzie nieokreślony. Można także z właściwej części proceduralnej odwoływać się do sekcji USE, ale sterowanie musi wrócić do właściwej części proceduralnej, bo w przeciwnym przypadku wynik działania programu byłby nie-

określony. Ważne jest to, że w sekcjach USE i procedurach przez nie wywoływanych nie można wykonywać instrukcji: OPEN, READ, WRITE, CLOSE. Sekcje USE mogą być segmentowane w ten sam sposób co zwykłe procedury.

Niestandardowe procedury błędu

Wykonanie niestandardowej procedury błędu jest zależne od wyniku działania standardowej procedury błędu:

- jeżeli po wykonaniu standardowej procedury błędu następuje powrót do programu, to procedura niestandardowa jest wykonywana bezpośrednio po tym powrocie,
- jeżeli standardowa procedura błędu powoduje zawieszenie pracy maszyny, to niestandardowa procedura jest wykonywana bezpośrednio po zlikwidowaniu przez operatora stanu zawieszenia,
- jeżeli standardowa procedura błędu powoduje przerwanie programu to niestandardowa procedura nie będzie wykonana.

Wykaz standardowych procedur błędu dla wejścia/wyjścia wraz z opisem ich czynności jest podany w Dodatku F (zob. także pkt 5.3.2).

W niestandardowych procedurach błędu może wystąpić zmienna BUFF, która odnosi się do bufora zbioru. Można więc w przypadku błędu zażądać wypisania zawartości bufora zbioru używając do tego instrukcji DISPLAY (zob. opis instrukcji DISPLAY). Jeśli w programie jest opisany więcej niż jeden zbiór, zmienna BUFF musi

być kwalifikowana nazwą zbioru, którego pole buforowe ma być wypisane.

Niestandardowe procedury metrykowe

Poniżej podano w jakim miejscu ciągu operacji standardowej procedury metrykowej wykonywane są niestandardowe procedury metrykowe w zależności od tego, które słowa kluczowe wystąpią w zdaniu USE.

$$\underline{\text{BEFORE}} \left\{ \begin{array}{c} \underline{\text{BEGINNING}} \\ \underline{\text{ENDING}} \end{array} \right\} \left\{ \begin{array}{c} \underline{\text{REEL}} \\ \underline{\text{FILE}} \end{array} \right\} \underline{\text{INPUT}}$$

Czynności:

- wczytanie metryki $\left\{ \begin{array}{cc} \text{początku} & \text{szpuli} \\ \text{końca}^{1)} & \text{zbioru} \end{array} \right\}$
- wykonanie procedury niestandardowej,
- wykonanie procedury standardowej, tj. sprawdzenie poszczególnych pozycji metryki,
- powrót do programu głównego.

$$\underline{\text{BEFORE}} \left\{ \begin{array}{c} \underline{\text{BEGINNING}} \\ \underline{\text{ENDING}} \end{array} \right\} \left\{ \begin{array}{c} \underline{\text{REEL}} \\ \underline{\text{FILE}} \end{array} \right\} \underline{\text{OUTPUT}}$$

Czynności:

- wykonanie procedury niestandardowej,

1) w przypadku zbiorów związanych z symbolicznym wejściem-wyjściem dotyczy to ogranicznika zbioru,

- wykonanie procedury standardowej, tj.

utworzenie metryki $\left\{ \begin{array}{ll} \text{początku} & \text{szpuli} \\ \text{końca} & \text{zbioru} \end{array} \right\}$

- wypisanie metryki zbioru,
- powrót do programu głównego.

AFTER $\left\{ \begin{array}{l} \underline{\text{BEGINNING}} \\ \underline{\text{ENDING}} \end{array} \right\} \left\{ \begin{array}{l} \underline{\text{REEL}} \\ \underline{\text{FILE}} \end{array} \right\} \underline{\text{INPUT}}$

Czynności:

- wykonanie procedury standardowej, tj. sprawdzenie poszczególnych pozycji metryki,
- wykonanie procedury niestandardowej,
- powrót do programu głównego.

AFTER $\left\{ \begin{array}{l} \underline{\text{BEGINNING}} \\ \underline{\text{ENDING}} \end{array} \right\} \left\{ \begin{array}{l} \underline{\text{REEL}} \\ \underline{\text{FILE}} \end{array} \right\} \underline{\text{OUTPUT}}$

Czynności:

- wykonanie procedury standardowej, tj.

utworzenie metryki $\left\{ \begin{array}{ll} \text{początku} & \text{szpuli} \\ \text{końca}^1) & \text{zbioru} \end{array} \right\}$

- wykonanie procedury niestandardowej,
- wypisanie metryki zbioru,
- powrót do programu głównego.

Jeżeli w zdaniu USE w wersji II występuje słowo INPUT (względnie OUTPUT), to niestandardowa procedura wykonywana jest dla wszystkich zbioru

1) zob. odnośnik na stronie poprzedniej

rów wejściowych (względnie wyjściowych), z wyjątkiem tych, którym w DATA DIVISION przypisano zwrot LABEL RECORD OMITTED.

W sekcji USE wolno się odwoływać do wszystkich zmiennych opisanych w DATA DIVISION, do stałych figuratywnych, nazwy rejestru TALLY, zmiennych zastrzeżonych: BUFF (która jest pozycją znakową), HARDWARE oraz HARDWAREi (obie typu FIXED), a ponadto do zmiennych będących nazwami pozycji metrykowych. Są to następujące zmienne:

<u>Zmienna</u>	<u>własność</u>	<u>występuje w metryce</u>
SECT	FIXED	początku - dla zbiorów na taśmie magnetycznej,
IDENT	PICTURE 'X(18)'	początku - dla dowolnych zbiorów,
CREAT	FIXED	początku - dla zbiorów na taśmie magnetycznej,
GENER	FIXED	początku - dla zbiorów na taśmie magnetycznej,
EXPIR	FIXED	początku - dla zbiorów na taśmie magnetycznej,

Do pozycji metryki końca zbioru programista nie ma dostępu.

4.4.7.5. Zdanie COPY

Funkcją zdania COPY jest włączanie do programu źródłowego fragmentów przechowywanych w bibliotece systemu. Zdanie COPY ma następującą postać ogólną:

COPY <nazwa-biblioteczna>

<nazwa-biblioteczna> jest identyfikatorem zawierającym najwyżej cztery znaki.

Zdanie COPY może występować w każdej z części: ENVIRONMENT, DATA oraz PROCEDURE, zgodnie z poniższymi formatami:

w ENVIRONMENT DIVISION:

SPECIAL..NAMES . COPY<nazwa-biblioteczna>.
FILE..CONTROL. COPY<nazwa-biblioteczna>.
I..O..CONTROL. COPY<nazwa-biblioteczna>.

w DATA DIVISION:

FD<nazwa-zbioru>COPY<nazwa-biblioteczna>.
SD<nazwa-zbioru>COPY<nazwa-biblioteczna>.
O1<nazwa-dokumentu>COPY<nazwa-biblioteczna>.

w PROCEDURE DIVISION:

<nazwa-procedury>. COPY<nazwa-biblioteczna>.

Kopiuwany zbiór biblioteczny jest fragmentem programu napisanym w języku COBOL. W zbiorze włączonym do PROCEDURE DIVISION programu źródłowego można używać języka WSTAWEK.

Zbiór biblioteczny włączany jest do programu źródłowego wyłącznie na miejsce zdania COPY (łącznie z kończącą go kropką). Zbiór ten musi być taki, aby po jego włączeniu otrzymany program źródłowy był syntaktycznie poprawny.

W tekście stanowiącym zbiór biblioteczny nie może występować zdanie COPY.

Jeżeli w trakcie tłumaczenia drukowany jest tabulogram programu źródłowego (zob. opis dyrektywy TAB), to w tym tabulogramie obok zdania COPY umieszczana jest zawartość odpowiadającego mu zbioru bibliotecznego.

Sposób redagowania i tworzenia zbiorów bibliotecznych dla zdania COPY opisany jest w pkt. 5.1 "Zbiory biblioteczne".

4.4.7.6. Zdanie END

Zdanie END kończy część DECLARATIVES albo jest ostatnim zdaniem w programie źródłowym.

Postać ogólna

$$\underline{\text{END}} \quad \left\{ \begin{array}{l} \underline{\text{DECLARATIVES}} \\ \underline{\text{PROGRAM}} \end{array} \right\}$$

Zdanie END DECLARATIVES kończy część DECLARATIVES.

Zdanie END PROGRAM oznacza fizyczny koniec zapisu programu źródłowego.

4.4.8. INSTRUKCJE JĘZYKA COBOL

Alfabetyczny spis instrukcji języka COBOL podany jest w Dodatku B.

4.4.8.1. Instrukcje związane z wejściem-wyjściem

Są to instrukcje OPEN, READ, WRITE, CLOSE odnoszące się do zbiorów danych oraz instrukcje ACCEPT i DISPLAY odnoszące się do pojedynczych zmiennych.

4.4.8.1.1. Instrukcja OPEN

Instrukcja OPEN rozpoczyna przetwarzanie zbiorów danych, dla zbiorów na taśmie magnetycznej

powoduje ustawienie taśmy na początku zbioru. Instrukcja OPEN sprawdza metrykę początku zbioru dla zbioru wejściowego, a tworzy i wypisuje metrykę dla zbioru wyjściowego.

Postać ogólna instrukcji OPEN podana jest we wkładce III na końcu rozdziału 4.

Przykłady

OPEN INPUT EWIDENCJA, DANE OUTPUT WYKAZ.

OPEN INPUT RACHUNKI..PROSTE.

W instrukcji OPEN musi wystąpić jedna z fraz: INPUT lub OUTPUT, a jeśli występują obie, to kolejność tych fraz jest dowolna.

Instrukcja OPEN musi być wykonana dla wszystkich zbiorów przetwarzanych w programie, z wyjątkiem zbiorów sortowania, do których instrukcja ta nie może się odnosić.

Instrukcja OPEN dla określonego zbioru musi być w programie wykonana przed wykonaniem w programie instrukcji READ lub WRITE odnoszącej się do tego zbioru.

Następna instrukcja OPEN nie może być dla zbioru wykonana zanim dla tego zbioru nie zostanie wykonana instrukcja CLOSE.

Instrukcja OPEN nie powoduje wczytania czy wypisania pierwszego dokumentu zbioru (do tego celu służą odpowiednio instrukcje READ oraz WRITE). Ustawia ona nośnik, z którym związany jest zbiór, po metryce początku tego zbioru.

Jeżeli otwierany zbiór związany jest z symbolicznym wejściem-wyjściem, tj. w deklaracji SELECT, opisującej ten zbiór, występuje fraza

INPUT<całkowita> bądź OUTPUT<całkowita> to instrukcja OPEN nie dołącza tego symbolicznego wejścia-wyjścia.

W trakcie wykonywania instrukcji OPEN dla zbiorów wejściowych sprawdzana jest metryka początku zbioru, a dla zbiorów wyjściowych - wypisywana jest metryka początku zbioru. Wartości poszczególnych pozycji w metryce początku zbioru określane są w programie za pomocą zwrotu VALUE OF (zob. opis zbioru w rozdziale DATA DIVISION):

- gdy zbiór związany jest z symbolicznym wejściem-wyjściem, wówczas zwrotem VALUE OF wolno określać jedynie wartość pozycji IDENT; jedynie pozycja IDENT sprawdzana jest w trakcie wykonania instrukcji OPEN,
- gdy zbiór związany jest z taśmą magnetyczną, wówczas zwrotem VALUE OF można określić wartości pozycji IDENT, SECUR oraz SECT,
- gdy w zwrocie VALUE OF programista nie określi wartości którejkolwiek z wymienionych wyżej pozycji metrykowych, wówczas pozycjom tym nadawana jest wartość domyślna.

Jeżeli zbiór jest związany z taśmą magnetyczną i w opisie tego zbioru użyto zwrotu LABEL RECORD z frazą WITH GENER, to:

- w czasie otwierania zbioru jako wejściowego następuje zawieszenie maszyny i użytkownik musi wprowadzić z monitora numer generacji, który zostanie porównany z numerem generacji zbioru,

- gdy obie te liczby nie są sobie równe, wówczas użytkownik musi pośrodkowo wprowadzić numer generacji z monitora. Czynność ta musi być powtarzana tak długo, aż obie te liczby będą równe; w czasie otwierania zbioru dla wyjścia następuje zawieszenie maszyny i numer generacji, wprowadzony przez użytkownika z monitora, zostaje wniesiony do odpowiedniej pozycji metryki; numer generacji musi być liczbą całkowitą pięciocyfrową zakończoną gwiazdką '*' (np. 00008*).

Gdy zbiór związany jest z taśmą magnetyczną, wówczas czynności wykonywane w ramach jego otwarcia zależą od tego, czy nazwa tego zbioru występuje w danym programie w deklaracji MULTIPLE, czy nie.

Dla zbioru wejściowego, przed wczytaniem i sprawdzeniem metryki początku zbioru, wykonywane są czynności:

- jeżeli nazwa zbioru nie występuje w deklaracji MULTIPLE, to każda szpula taśmy związana ze zbiorem musi być przez operatora maszyny ustawiona na początku; powinna znajdować się tam metryka początku otwieranego zbioru,
- gdy nazwa zbioru występuje w deklaracji MULTIPLE, wówczas metryka ta szukana jest na szpuli taśmy w kierunku rosnących numerów bloków.

Dla zbioru wyjściowego przed utworzeniem i wypisaniem metryki początku zbioru wykonywane są czynności:

- jeżeli nazwa zbioru nie występuje w deklaracji MULTIPLE, to szpula założona na pierwszej stacji zbioru¹⁾ powinna być ustawiona na metryce końca zapisu albo metryce początku dowolnego zdezaktualizowanego zbioru; metryka początku otwieranego zbioru wpisywana jest potem na miejsce tej metryki,
- gdy nazwa zbioru występuje w deklaracji MULTIPLE, wówczas szpula, na której zbiór ma być zapisany, jest przeszukiwana w kierunku rosnących numerów bloków - w celu znalezienia bloku zawierającego metrykę końca zapisu albo metrykę początku dowolnego zdezaktualizowanego zbioru; na miejsce takiego bloku wypisywana jest potem metryka początku otwieranego zbioru.

Jeżeli zbiór jest wejściowy i w opisie zbioru w DATA DIVISION przypisano mu zwrot LABEL RECORD IS OMITTED, to poprawność metryki początku zbioru nie jest sprawdzana, mimo że dla zbiorów związanych z taśmą magnetyczną metryka ta jest wczytywana.

W trakcie wykonywania instrukcji OPEN sprawdza się czy pole zbioru nie jest zajęte przez inny zbiór (dla opcji SAME AREA).

Jeżeli w programie występuje zdanie USE, odnoszące się do opracowywanego zbioru, to wykonywana jest odpowiednia niestandardowa procedura metrykowa (porównaj opis zdania USE).

4.4.8.1.2. Instrukcja READ

Wykonanie instrukcji READ udostępnia do przetwarzania kolejny dokument zbioru wejściowego.

¹⁾ tj. stacji, która wymieniona jest w deklaracji SELECT danego zbioru jako pierwsza.

Instrukcja READ ma następującą postać ogólną:

READ<nazwa-zbioru>RECORD AT END GO TO<nazwa-procedury>

Przykłady

READ DANE..WEJSCIOWE END GO TO KONIEC.

READ RACHUNKI AT END GO TO OBLICZENIA..ZBIORCZE.

Przed wykonaniem pierwszej instrukcji READ odnoszącej się do pewnego zbioru musi być wykonana dla tego zbioru instrukcja OPEN w wersji INPUT.

Jeżeli w trakcie wykonywania instrukcji READ zostanie wykryty ogranicznik zbioru albo metryka końca zbioru, to nie pobiera się kolejnego dokumentu zbioru, lecz sterowanie przekazywane jest do procedury, której nazwa wymieniona jest w frazie AT END tej instrukcji. Po wykonaniu skoku określonego we frazie AT END, do danego zbioru nie może się odnosić instrukcja READ, chyba że przed nią zostaną wykonane instrukcje CLOSE oraz OPEN.

Każdy kolejny dokument zbioru jest wczytywany zawsze do tego samego pola dokumentów również wtedy, gdy w zbiorze występują dokumenty kilku typów.

Jeżeli dokument wczytywany jest ze zbioru związanego z symbolicznym wejściem, to gdy ten zbiór jest pozycyjny, do rejestru specjalnego TALLY posyłana jest liczba całkowita równa liczbie znaków wczytanego dokumentu; gdy zbiór jest niepozycyjny, do rejestru TALLY posyłana

jest liczba całkowita określająca typ wczytanego dokumentu.

Sposób czytania dokumentów zbioru niepozycyjnego zależy od tego jak programista określa bufor zbioru. Gdy bufor zbioru równy jest największemu co do długości dokumentowi (tj. bufor nie jest opisany w programie zwrotem BLOCK), wówczas wczytywany jest od razu jeden dokument. Gdy natomiast programista opisze w programie bufor zwrotem BLOCK CONTAINS całkowita CHARACTERS, wówczas:

- jeżeli wczytywany dokument mieści się w buforze długości <całkowita>, to jest on wczytywany w całości, po czym następuje jego przekształcenie na postać wewnętrzną,
- jeżeli dokument jest dłuższy od buforu, to jest on wczytywany częściami równymi co do długości buforowi; każda taka część jest przekształcana na postać wewnętrzną zgodnie z opisem wczytywanego dokumentu i posyłana na kolejne miejsce w polu dokumentu; proces ten odbywa się tak długo, aż na nośniku zostanie napotkany ogranicznik dokumentu.¹⁾

Przy obydwu wymienionych sposobach czytania zbioru niepozycyjnego, w ramach wykonania jednej instrukcji READ, udostępniany jest zawsze jeden kolejny dokument.

Jeżeli dokument wczytywany jest ze zbioru związanego z taśmą magnetyczną, to do rejestru TA-

1) Czytanie dokumentu częściami jest możliwe tylko dla dokumentów ze znacznikami - inaczej sygnalizowany jest błąd (zob. opis technik czytania zbiorów niepozycyjnych).

LLY posyłana jest liczba całkowita równa liczbie słów zajmowanych przez ten dokument w polu dokumentu. Gdy dokument jest dłuższy od pola dokumentu, wówczas do tego pola posyłana jest tylko część dokumentu mieszcząca się w polu dokumentu (pozostała część dokumentu jest pomijana w dalszym przetwarzaniu), a do rejestru TALLY posyłana jest liczba całkowita równa długości pola dokumentu.

Jeżeli zbiór jest związany z taśmą magnetyczną i w trakcie czytania zostanie wykryty zbliżający się koniec szpuli, to w ramach instrukcji READ wykonywane są następujące czynności:

- standardowa wymiana szpuli (porównaj opis instrukcji CLOSE),
- pobranie pierwszego dokumentu z następnej szpuli.

Jeżeli w trakcie czytania zostanie wykryty błąd danych albo wada urządzenia i do danego zbioru odnosi się zdanie USE AFTER STANDARD ERROR PROCEDURE, to wykonanie niestandardowej procedury błędu, zadanej przez programistę, przebiega zgodnie z regułami podanymi w opisie zdania USE.

4.4.8.1.3. Instrukcja WRITE

Instrukcja WRITE powoduje przesłanie dokumentu do zbioru wyjściowego. Instrukcja WRITE ma następującą postać ogólną:

WRITE <nazwa-dokumentu>

Przykłady

WRITE KARTA..PRACY .

WRITE RW

Przed wykonaniem pierwszej instrukcji WRITE dla danego zbioru, musi być dla tego zbioru wykonana instrukcja OPEN w wersji OUTPUT. Nazwa dokumentu jest to nazwa opisana w sekcji zbiorów w DATA DIVISION z numerem poziomu 1; dokument ten nie może należeć do zbioru sortowania. Mimo wykonania instrukcji WRITE, wypisywany dokument jest w dalszym ciągu dostępny w polu dokumentu, dopóki do tego pola nie zostanie przesłana inna informacja.

Jeżeli zbiór, do którego przesyłane są dokumenty, jest związany z taśmą magnetyczną i w czasie pisania zostanie wykryty zbliżający się koniec szpuli, to nastąpi standardowa wymiana szpuli (porównaj opis instrukcji CLOSE) i bieżący dokument zostanie zapisany na nowej szpuli.

Wykrycie błędu zapisu w trakcie pisania dokumentu powoduje przejście do standardowej procedury błędu. Jeżeli do danego zbioru odnosiło się zdanie USE definiujące niestandardową procedurę błędu, to zostanie ona wykonana zgodnie z obowiązującymi regułami (zob. opis zdania USE).

4.4.8.1.4. Instrukcja CLOSE

Instrukcja CLOSE kończy przetwarzanie zbiorów lub kolejnych szpul zbiorów wejściowych i wyjściowych oraz może powodować nieodwinięcie szpul względnie zablokowanie zbioru.

Po wykonaniu instrukcji CLOSE dany zbiór uważany jest za "zamknięty", tj. nie można wykonywać dla niego instrukcji READ lub WRITE, chyba że zostanie ponownie "otwarty" za pomocą instrukcji OPEN.

Postać ogólna instrukcji CLOSE podana jest we wkladce III na końcu rozdziału.

Przykłady

CLOSE DANE, EWIDENCJA.

CLOSE KWITY WITH NO REWIND.

<nazwa-zbioru> użyta w instrukcji CLOSE nie może być nazwą zbioru sortowania.

Standardowe zamknięcie zbioru, tj. użycie instrukcji CLOSE bez fraz REEL, LOCK, NO REWIND powoduje wykonanie niżej opisanych czynności.

Zbiory wejściowe

Jeżeli zbiór jest ustawiony na końcu, tj. została wykonana fraza AT END instrukcji READ odnoszącej się do tego zbioru i zbiór jest metrykowany, to metryka zostaje wczytana, sprawdzane są poszczególne jej pozycje i zwalniane jest pole tego zbioru. Zwolnienie pola zbioru oznacza, że można już otworzyć inny zbiór, który z danym zbiorem był wymieniony w tym samym zwrocie SAME AREA.

Ponadto, jeżeli w programie występuje odpowiednie zdanie USE odnoszące się do tego zbioru, to przed (dla USE BEFORE ENDING FILE ON INPUT) lub po (dla USE AFTER ENDING FILE ON INPUT) sprawdzeniu pozycji metryki końca wykonywana jest niestandardowa procedura, zadana przez programistę. Po wy-

konaniu tych czynności zarówno dla zbiorów związanych z taśmą magnetyczną jak i nośnikiem papierowym, zwalniane jest pole zbioru. Jeśli zbiór nie posiada metryk lub jeśli nie wykonano dla niego frazy AT END, to metryka końca nie jest sprawdzana. Jeżeli zamykany zbiór związany jest z symbolicznym wejściem, to instrukcja CLOSE nie odłącza tego wejścia.

Zbiory wyjściowe

Gdy zbiór danych związany jest z taśmą magnetyczną, to tworzona jest i wypisywana metryka końca zbioru oraz metryka końca zapisu. Jeżeli w programie wystąpiło odpowiednie zdanie USE odnoszące się do tego zbioru, to przed utworzeniem i wypisaniem metryki końca zbioru (dla USE BEFORE ENDING FILE ON OUTPUT) lub po utworzeniu metryki końca zbioru lecz przed wypisaniem (dla USE AFTER ENDING FILE ON OUTPUT), wykonywana jest niestandardowa procedura zadana przez programistę.

Dla zbiorów związanych z nośnikiem papierowym wypisywany jest tylko ogranicznik zbioru. Po wykonaniu tych czynności zarówno dla zbiorów związanych z taśmą magnetyczną jak i nośnikiem papierowym, zwalniane jest pole zbioru. Jeśli zamykany zbiór związany jest z symbolicznym wyjściem, to instrukcja CLOSE nie odłącza tego wyjścia.

Jeżeli zbiór (wejściowy czy wyjściowy) zapisany jest na taśmie magnetycznej, to na zakończenie czynności związanych z wykonaniem dla tego

zbioru instrukcji CLOSE, szpula taśmy magnetycznej odwijana jest do początku.

Frazy REEL, LOCK, NO REWIND odnoszą się wyłącznie do zbiorów związanych z taśmą magnetyczną. Poniżej zostaną omówione poszczególne wersje instrukcji CLOSE z użyciem tych fraz.

CLOSE WITH LOCK - prócz standardowych czynności zamknięcia zbioru następuje blokada tego zbioru, to znaczy zbiór nie może być ponownie otwarty w tym samym programie.

CLOSE WITH NO REWIND - po standardowym zamknięciu zbioru szpula nie jest odwijana do początku. Fraza ta jest szczególnie przydatna w przypadku szpul, na których znajduje się kilka zbiorów (MULTI-FILE REEL).

Po zamknięciu opracowanego zbioru można otwierać następny, zapisany na tej samej szpuli.

CLOSE REEL - zamknięcie kolejnej szpuli zbioru zapisanego na kilku szpulach czyli zbioru wieloszpulowego. Wykonywane są niżej podane czynności.

Zbiory wejściowe

- Sprawdzenie metryki końca szpuli (o ile zbiór jest metrykowany),
- odwinięcie zamykanej szpuli do początku,
- otwarcie szpuli kontynuacyjnej.

Jeżeli dany zbiór wykorzystuje tylko jedną stację taśmy magnetycznej lub na żadnej ze stacji przypisanych temu zbiorowi nie ma szpuli kontynuacyjnej, to po odpowiednim monicie

praca maszyny jest zawieszana. Po założeniu odpowiedniej szpuli można pracę kontynuować. W związku z tym należy uważać, aby nie użyć tej wersji instrukcji CLOSE do zbioru wejściowego zapisanego na jednej szpuli lub do ostatniej szpuli zbioru wieloszpułowego gdyż nastąpi poszukiwanie szpuli kontynuacyjnej, a jej brak uniemożliwi powrót do normalnej pracy programu.

Jeżeli w programie do danego zbioru odnosi się odpowiednie zdanie USE, to przed (USE BEFORE ENDING REEL ON INPUT) lub po (USE AFTER ENDING REEL ON INPUT) sprawdzeniu metryki końca szpuli i przed (USE BEFORE BEGINNING REEL ON INPUT) lub po (USE AFTER BEGINNING REEL ON INPUT) sprawdzeniu metryki początku zbioru na szpuli kontynuacyjnej, wykonywana jest procedura niestandardowa zadana przez programistę.

Zbiory wyjściowe

- Wypisanie metryki końca szpuli,
- odwiniecie zamykanej szpuli do początku,
- wypisanie metryki początku szpuli kontynuacyjnej.

Jeżeli dany zbiór wykorzystuje tylko jedną stację taśmy magnetycznej lub na żadnej ze stacji przypisanych temu zbiorowi nie ma szpuli, która mogłaby być szpulą kontynuacyjną, to po odpowiednim monicie praca maszyny jest zawieszana. Po założeniu odpowiedniej szpuli wykonywanie programu jest wznowiane.

Jeżeli w programie do danego zbioru odnosi się odpowiednie zdanie USE, to przed (USE BEFORE ENDING

REEL ON OUTPUT) utworzeniem lub po (USE AFTER ENDING REEL ON OUTPUT), utworzeniu, ale przed wypisaniem metryki końca szpuli wykonywana jest procedura zadana przez programistę. Niestandardowe procedury zadane zdaniem USE, odnoszące się do metryki początku szpuli kontynuacyjnej, są wykonywane (USE BEFORE BEGINNING REEL ON OUTPUT) przed utworzeniem tej metryki lub (USE AFTER BEGINNING REEL ON OUTPUT) po utworzeniu lecz przed wypisaniem metryki.

CLOSE REEL WITH NO REWIND - wszystkie czynności jak przy wersji CLOSE REEL z tym, że zamykana szpula nie jest odwijana.

CLOSE REEL WITH LOCK - ta wersja jest niedozwolona gdyż zablokowany zostanie cały zbiór, co uniemożliwi otwarcie szpuli kontynuacyjnej.

4.4.8.1.5. Instrukcja ACCEPT

Instrukcja ACCEPT służy do wczytywania elementarnych jednostek danych z określonego wejścia symbolicznego do pól zadanych pozycji.

Postać ogólna instrukcji ACCEPT podana jest we wkladce III na końcu rozdziału.

Przykłady

ACCEPT X(5,2), TYTUŁ FROM INPUT 3.

ACCEPT WSKAŹNIK..1.

Symboliczne wejście <INPUT całkowita> musi być zdefiniowane w czołówce programu, musi zawierać dekodery na KWS i być dołączone we wstawce - prócz wejścia 0, które jest dołączane automatycznie przez system operacyjny (zob.pkt 5.3.5).

Jeśli w instrukcji ACCEPT pominięto frazę FROM INPUT<całkowita>, to jednostki danych będą pobierane z urządzenia zdefiniowanego w programie jako wejście zerowe.

Jednostki danych na nośniku zewnętrznym mogą być liczbami ze znakiem lub ciągami znaków w apostrofach. Każda jednostka musi być zakończona średnikiem, np.

+10; 'TABELA'; -256;

Liczba może być wczytana do pozycji obliczeniowej (tj. typu FIXED, FIXED2, FLOAT) lub opisanej obrazem liczbowym, a ciąg znaków - do pozycji opisanej obrazem tekstowym. Maksymalna długość ciągów wynosi 80 znaków, a ograniczające tekst apostrofy nie są wliczane do długości ciągu. Ciągi dłuższe są skracane przez pominięcie znaków z prawej strony za znakiem osiemdziesiątym. Znaki spacji występujące przed lub za średnikiem kończącym jednostkę danych - nie mają znaczenia.

Jednostka danych na nośniku może być pusta, tj. przed pierwszym średnikiem lub między dwoma średnikami może nie wystąpić żadna informacja. W tym przypadku wartość pozycji, do której miała być wczytana ta jednostka - nie ulegnie zmianie.

Argumentem instrukcji ACCEPT mogą być nazwy zmiennych zastrzeżonych HARDWARE i HARDWAREi, z tym, że wolno do nich wczytywać tylko liczby całkowite, dodatnie.

4.4.8.1.6. Instrukcja DISPLAY

Instrukcja DISPLAY służy do wypisywania na określone symboliczne wyjście tekstów lub wartości pozycji, których lista podana jest w instrukcji. Postać ogólna instrukcji DISPLAY podana jest we wkładce III na końcu rozdziału.

Przykład

```
DISPLAY '.4SPALFA.SP=.SP', ALFA, (5) SPACE,  
      'BETA.SP=.SP', BETA, '.4LF' OUTPUT 2.
```

Symboliczne wyjście (OUTPUT<całkowita>) musi być zdefiniowane w czołówce programu, musi zawierać dekodery na KWS i być dołączone we wstawce (porównaj punkt 5.3.5). Jeśli w instrukcji DISPLAY pominięto frazę UPON OUTPUT całkowita, to wypisywanie odbywa się na urządzeniu zdefiniowanym w danym programie jako wyjście symboliczne zero.

Każda <nazwa-jednostki-danych> musi być nazwą pozycji elementarnej typu znakowego, tj. opisanej zwrotem PICTURE z dowolnym obrazem. <wyrażenie-arytmetyczne> określa ile razy wartość argumentu ma być wypisana (w przykładzie (5) SPACE określa pięć spacji). Brak wyrażenia arytmetycznego oznacza wypisanie argumentu raz. Jeśli wartość <wyrażenia-arytmetycznego> jest równa zero lub ujemna, to instrukcja DISPLAY dla tego argumentu jest pusta. Stałe figuratywne HIGH..VALUE, LOW..VALUE i SPACE, które mogą być argumentami instrukcji DISPLAY są odpowiednimi tekstami o długości 1.

Instrukcja DISPLAY nie wypisuje żadnych dodatkowych znaków prócz wartości argumentów.

Jeżeli argumentem instrukcji DISPLAY jest zmienna zastrzeżona BUFF - wówczas przy wypisywaniu zawartości bufora zbioru poszczególne pozycje dokumentu oddzielane są spacjami.

Zmienna BUFF może być argumentem instrukcji DISPLAY tylko w części DECLARATIVES w sekcji USE definiującej niestandardową procedurę błędu (zob. opis zdania USE).

4.4.8.2. Instrukcje obliczeniowe i przes- łania

W języku COBOL dla maszyny ZAM 41 istnieje tylko jedna instrukcja obliczeniowa - COMPUTE, jedna instrukcja przesłania - MOVE oraz instrukcja EXAMINE, za pomocą której można obliczyć liczbę wystąpień wskazanego znaku i/lub zastąpić go innym.

4.4.8.2.1. Instrukcja COMPUTE

Instrukcja COMPUTE powoduje wyliczenie wartości wyrażenia arytmetycznego i przesłanie tej wartości do pola pozycji występującej z lewej strony znaku równości.

Postać ogólna instrukcji COMPUTE podana jest we wkładce III na końcu tego rozdziału.

Przykłady

```
COMPUTE SREDNIA = A (5,I + J) * W1 + B (J)
                *(W2 / (W1 + W2)).
```

COMP TX ROUNDED = 3.14*RA.

Wyrażenie-arytmetyczne musi być zbudowane zgodnie z regułami podanymi w 4.4.6.1. Wyrażenia arytmetyczne.

<Nazwa-jednostki-danych>, występująca po lewej stronie znaku równości, musi być opisana zwrotem FIXED, FIXED2 lub FLOAT, obrazem liczbowym lub obrazem liczbowym wydawniczym. Przesłanie wyniku do pola opisanego zwrotem FIXED, FIXED2, FLOAT odbywa się zawsze z zaokrągleniem, a do pola opisanego obrazem tylko wtedy, gdy wystąpiła fraza ROUNDED. Zaokrąglenie takie odbywa się następująco: przed przesłaniem do pozycji wynikowej, po przekształceniu wyniku na postać dziesiętną, dodaje się jedność do najmniej znaczącej cyfry wyniku, gdy pierwsza odcięta cyfra wyniku jest większa od 5.

Użycie frazy ON SIZE ERROR powoduje sprawdzenie czy obliczenie wartości <wyrażenia-arytmetycznego> nie spowodowało nadmiaru - tj. ustawienia wskaźnika nadmiaru maszyny oraz czy wartość <wyrażenia-arytmetycznego> mieści się w polu wynikowym.

Jeżeli wystąpiła jedna z wyżej wymienionych okoliczności lub obie, to sterowanie zostaje przekazane do procedury, której nazwa została wymieniona we frazie ON SIZE ERROR, a zawartość pola wynikowego nie ulega zmianie.

4.4.8.2.2. Instrukcja MOVE

Instrukcja MOVE przesyła wartości określonej jednostki danych do jednej lub więcej pozycji. Postać ogólna instrukcji MOVE znajduje się we wkładce III na końcu tego rozdziału.

Przykłady

MOVE ZERO TO A.

MOVE 1 TO WSKAZNIK, X, Y.

MOVE X TO Y.

Nazwy pozycji, których wartości mają być przesłane, tj. występują w instrukcji MOVE między słowami zastrzeżonymi MOVE i TO, nazywamy pozycjami źródłowymi a nazwy jednostek danych, do których wartości będą przesłane, tj. stojące za słowem zastrzeżonym TO - nazywamy pozycjami docelowymi. Jak widać z postaci ogólnej w instrukcji MOVE może być tylko jedna pozycja źródłowa, a pozycji docelowych może być kilka.

Stałe figuratywne HIGH..VALUE, LOW..VALUE, SPACE oraz nazwa pozycji metrykowej IDENT są pozycjami znakowymi. Stała figuratywna ZERO, nazwa rejestru TALLY i nazwy pozycji metrykowych SECT, CREAT, GENER oraz EXPIR są pozycjami obliczeniowymi. Jak pamiętamy, pozycje metrykowe mogą wystąpić w instrukcji MOVE tylko w części DECLARATIVES.

Instrukcja MOVE nie zmienia wartości pozycji źródłowej.

Przesłaniem elementarnym będziemy nazywali każde przesłanie, w którym pozycja źródłowa i pozycja docelowa są elementarne.

W przypadku przesłań elementarnych obowiązują następujące reguły:

- jeśli pozycja docelowa jest tekstowa, to przesłanie zachodzi zgodnie z regułami podanymi w opisie zwrotu JUSTIFIED,

- jeśli pozycja docelowa jest znakowa typu liczbowego lub liczbowego wydawniczego, to w trakcie przesłania dokonuje się adiustacji w stosunku do położenia kropki dziesiętnej, z ewentualnym uzupełnieniem zerami - gdy pozycja źródłowa jest krótsza od docelowej; jeśli w obrazie pozycji docelowej nie występuje symbol S, to przesyłana jest wartość bezwzględna pozycji źródłowej; jeśli pozycja źródłowa, po adiustacji w stosunku do położenia kropki dziesiętnej w pozycji docelowej, ma cyfry znaczące nie mieszczące się w polu pozycji docelowej z lewej lub prawej strony kropki, to te cyfry są obcinane,
- jeśli pozycja docelowa jest obliczeniowa, to wartość pozycji źródłowej jest tłumaczona na postać binarną i w tej postaci posyłana do pozycji docelowej; w przypadku, gdy wartość pozycji źródłowej jest po przetłumaczeniu większa od maksymalnej wartości mieszczącej się w pozycji docelowej - przesyłana wartość będzie nieokreślona,
- jeśli pozycja docelowa jest tekstowa wydawnicza, to przesłanie następuje zgodnie z regułami wydawnictwa,
- jeżeli trzeba, to przy przesłaniach dokonuje się przekształcenia danych z jednej reprezentacji wewnętrznej na inną (np. z postaci `FIXED` na `FLOAT`).

Przy przesłaniach elementarnych obowiązują następujące ograniczenia:

- nie wolno przysyłać pozycji liczbowej wydawniczej, tekstowej wydawniczej i tekstowej do pozycji obliczeniowej, liczbowej lub liczbowej wydawniczej,

TABELA PRZESŁAŃ ELEMENTARNYCH W INSTRUKCJI MOVE

Typ pozycji docelowej	Typ pozycji źródłowej	FIXED	FIXED2	FLOAT	liczbo- wa	teksto- wa	liczbowa wydawni- cza	tekstowa wydawni- cza
FIXED		X	X	X	X	B	X	B
FIXED2		X	X	X	X	B	X	B
FLOAT		X	X	X	X	B	X	B
liczbowa		X	X	X	X	X	X	X
tekstowa		B	B	B	B	X	B	X
liczbowa-wydawnicza		B	B	B	B	X	B	X
tekstowa-wydawnicza		B	B	B	B	X	B	X

X - oznacza przesłanie poprawne
B - oznacza przesłanie błędne

• nie wolno przesyłać pozycji obliczeniowej do pozycji tekstowej lub tekstowej wydawniczej.

Załączona tabelka pokazuje, które przesłania elementarne są dozwolone.

Przesłania, w których pozycja źródłowa i/lub docelowa jest nieelementarna nazywane przesłaniami grupowymi.

Przesłanie grupowe odbywa się tak, jak gdyby pozycja źródłowa i docelowa, były pozycjami tekstowymi.

4.4.8.2.3. Instrukcja EXAMINE

Instrukcja EXAMINE liczy wystąpienia określonego znaku w jednostce danych i/lub zastępuje określone znaki tej jednostki danych innymi znakami.

Postać ogólna instrukcji EXAMINE została umieszczona we wkładce III na końcu tego rozdziału, a przykłady na następnej stronie.

<nazwa-jednostki-danych> musi dotyczyć pozycji elementarnej, opisanej zwrotem PICTURE z dowolnym obrazem.

Znaki pozycji <nazwa-jednostki-danych> są opracowywane w kolejności od lewej do prawej.

W przypadku użycia wersji TALLYING do rejestru specjalnego TALLY posyłana jest:

- liczba znaków różnych od <znak-1>, występujących w badanej pozycji przed pierwszym znakiem <znak-1> - jeśli użyto frazy UNTIL FIRST,
- ilość znaków <znak-1>, występujących w badanej pozycji - jeśli użyto frazy ALL,

Przykłady:

Założmy, że zmienna X ma wartość 'KARTOTEKA-I'. Wtedy w wyniku wykonania:

instrukcji:	wartość zmiennej X będzie:	zaś wartość rejestru TALLY będzie:
EXAMINE X TALLYING UNTIL FIRST 'O';	bez zmian	4
EXAMINE X TALLYING ALL 'A';	bez zmian	2
EXAMINE X REPLACING ALL 'I' BY 'V';	'KARTOTEKA-V'	bez zmian

- ilość znaków <znak-1>, występujących w badanej pozycji przed pierwszym znakiem różnym od <znak-1> - jeśli użyto LEADING.

Jeśli obok wersji TALLYING użyto wersji REPLACING, wówczas wewnątrz badanej pozycji każdy <znak-1>, liczony w rejestrze TALLY, jest ponadto zamieniany na <znak-2>.

Instrukcja EXAMINE bez wersji TALLYING oznacza zamianę znaków <znak-3> na <znak-4> przy niezmienionej wartości TALLY.

Jak widać z postaci ogólnej w instrukcji EXAMINE mogą wystąpić w odpowiednich miejscach stałe figuratywne HIGH..VALUE, LOW..VALUE i SPACE. Są one rozumiane jako jeden znak o wartości reprezentowanej przez stałą figuratywną.

4.4.8.3. Instrukcje sterujące

W języku COBOL dla maszyny ZAM 41 istnieją następujące instrukcje sterujące: GO TO, IF, ALTER, PERFORM, STOP.

4.4.8.3.1. Instrukcja GO TO

Instrukcja GO TO przekazuje sterowanie do wymienionej w tej instrukcji procedury.

Postać ogólna instrukcji GO TO podana jest we wkładce III na końcu tego rozdziału.

Przykłady

GO TO B.

GO TO E1, E2, E3, E4 DEPENDING ON TALLY.

Każda <nazwa-procedury> występująca w instrukcji GO TO musi być nazwą sekcji lub paragra-

fu z PROCEDURE DIVISION. <nazwa-jednostki-danych> może być nazwą dowolnej jednostki elementarnej, obliczeniowej typu FIXED, lub inną wielkością języka COBOL typu FIXED (np. nazwą rejestru TALLY). Wartość tych pozycji musi być całkowita i nieujemna.

Instrukcja GO TO bez frazy DEPENDING przekazuje sterowanie do jednej tylko procedury, która może zostać zmieniona za pomocą instrukcji ALTER (zob. opis instrukcji ALTER).

Użycie frazy DEPENDING powoduje przekazanie sterowania do jednej z kolejno wymienionych procedur w zależności od wartości pozycji <nazwa-jednostki-danych>. Jeżeli wartość tej pozycji wynosi k, to sterowanie będzie przekazane do procedury, której nazwa jest wymieniona na miejscu k w instrukcji GO TO. Gdyby k było większe od liczby wymienionych procedur, ujemne lub równe zeru, instrukcja GO TO będzie pominięta przy wykonywaniu programu. Liczba nazw proceduralnych występujących w instrukcji GO TO z frazą DEPENDING - nie jest ograniczona. Jednak przy wymienianiu dużej liczby nazw należy pamiętać o ogólnym ograniczeniu długości zdań części proceduralnej (wynikającym z pojemności bufora wejściowego w translatorze).

4.4.8.3.2. Instrukcja IF

Instrukcja IF wylicza wartość podanego w niej warunku i zależnie od tego, czy jest on spełniony czy nie, przekazuje sterowanie w różne miejsca programu.

Postać ogólna instrukcji IF podana jest we wkładce III na końcu tego rozdziału.

Przykłady

IF A LESS B THEN GO TO S1 ELSE NEXT.

IF A EQUAL B THEN

IF C GREATER D THEN

IF D LESS 0 THEN GO TO X

ELSE COMPUTE D = 0; GO TO X

ELSE COMPUTE C = 0; GO TO X

ELSE GO TO X.

Pojęcie <warunek> opisane jest w punkcie 4.4.6.

Działanie instrukcji IF jest następujące:

- jeżeli <warunek> jest spełniony, to wykonywane są zdania napisane po słowie THEN a przed słowem ELSE,
- jeżeli warunek nie jest spełniony, to wykonywane są instrukcje napisane po słowie ELSE,
- po wykonaniu ciągu instrukcji po słowie THEN lub ELSE sterowanie przechodzi do następnego zdania po instrukcji IF, chyba że w tych ciągach wystąpiła instrukcja zmieniająca kolejność wykonywania programu np. instrukcja GO TO,
- użycie frazy NEXT SENTENCE powoduje przejście do następnej instrukcji po instrukcji IF. W ciągach instrukcji po słowach THEN, ELSE mogą występować (ale tylko jako ostatnie w ciągu) inne instrukcje IF. Jest to tzw. zagnieżdżanie instrukcji IF - jak w drugim przykładzie. Zagnieżdżone instrukcje IF odczytuje się w ten sposób, że licząc od lewej do prawej - kolejne napotkane słowo ELSE stanowi parę dla najbliższych poprzedzających

go słów IF i THEN nie posiadających pary. Oczywiście jest, że wszystkie słowa IF i THEN muszą mieć swoje pary w postaci słowa ELSE.

4.4.8.3.3. Instrukcja ALTER

Instrukcja ALTER powoduje dynamiczną zmianę argumentu instrukcji GO TO.

Instrukcja ALTER ma następującą postać ogólną:

ALTER <nazwa-paragrafu> TO <nazwa-procedury>

Przykład

```
X. GO TO A.  
:  
:  
Y.  
    ALTER X TO B.  
:  
A.  
:  
B.  
:  
:
```

W powyższym przykładzie instrukcja ALTER powoduje zmianę nazwy paragrafu A na B w instrukcji GO TO w paragrafie X.

Paragraf, którego nazwa występuje jako pierwszy argument instrukcji ALTER, musi być zawarty w PROCEDURE DIVISION i składać się tylko z instrukcji GO TO bez frazy DEPENDING. <nazwa-procedury> może być nazwą paragrafu lub sekcji zawartej w PROCEDURE DIVISION.

Instrukcja ALTER zawarta w segmencie wymiennym nie może dotyczyć instrukcji GO TO zawartej w innym segmencie wymiennym.

Jeżeli wykonano instrukcję ALTER dotyczącą instrukcji GO TO znajdującej się w segmencie wymiennym, to po każdorazowym sprowadzeniu segmentu wymiennego do pamięci operacyjnej ta instrukcja GO TO jest znowu w swojej pierwotnej postaci, tj. z niezmienionym argumentem.

4.4.8.3.4. Instrukcja PERFORM

Instrukcja PERFORM przekazuje sterowanie do określonej procedury lub ciągu procedur w celu wykonania ich określoną liczbę razy, po czym sterowanie przekazywane jest do instrukcji następnej po instrukcji PERFORM.

Postać ogólna instrukcji PERFORM podana jest we/wkładce III na końcu rozdziału.

Przykłady

PERFORM ITERACJE THRU KONIEC..ITERACJI UNTIL X-Y LESS 0.002.

PERFORM WYPELNIANIE..LISTY N TIMES.

<nazwa-procedury-1> i <nazwa-procedury-2> są nazwami sekcji lub paragrafów w PROCEDURE DIVISION.

Procedura <nazwa-procedury-1> musi dynamicznie występować w programie przed procedurą <nazwa-procedury-2>, tak aby można było wykonać program od procedury <nazwa-procedury-1> do procedury <nazwa-procedury-2>.

Pozycja <nazwa-jednostki-danych> musi być typu FIXED.

Pojęcie <warunek> opisane jest w punkcie 4.4.6.2.

Instrukcja PERFORM przekazuje sterowanie do pierwszej instrukcji procedury <nazwa-procedury-1>. W przypadku, gdy nie użyto ani frazy TIMES, ani frazy UNTIL, automatyczny powrót sterowania do instrukcji następnej po instrukcji PERFORM następuje:

- po wykonaniu ostatniej instrukcji procedury <nazwa-procedury-1> - jeśli nie użyto frazy THRU,
- po wykonaniu ostatniej instrukcji procedury <nazwa-procedury-2> - jeśli użyto frazy THRU.

Automatyczny powrót może nie nastąpić jeśli w zakresie instrukcji PERFORM (zob. opis zdania EXIT) występuje instrukcja przekazująca sterowanie poza ten zakres.

W procedurach wywoływanych za pomocą instrukcji PERFORM mogą być zawarte inne instrukcje PERFORM z następującymi ograniczeniami:

- dopuszczalne jest co najwyżej 8-krotne zagnieżdżenie instrukcji PERFORM,
- ciąg procedur wywoływany przez zagnieżdżoną instrukcję PERFORM musi być albo rozłączny z ciągiem procedur wywołanych przez poprzednio zapoczątkowaną instrukcję PERFORM, albo zawarty wewnątrz tego ciągu procedur.

Procedury wywoływane za pomocą instrukcji PERFORM mogą poza tym być także wykonywane w porządku sekwencyjnym.

W przypadku, gdy w instrukcji PERFORM użyto frazy TIMES, ciąg procedur <nazwa-procedury-1> do <nazwa-procedury-2> jest wykonywany tyle razy ile wynosi wartość pozycji <nazwa-jednostki-danych> lub <całkowita> w chwili zapoczątko-

wania instrukcji PERFORM; jeżeli ta wartość wynosi zero lub jest ujemna - instrukcja PERFORM nie jest wykonywana.

W trakcie wykonywania instrukcji PERFORM z frazą UNTIL sprawdzana jest prawdziwość<warunku>. Gdy <warunek> nie jest spełniony, wówczas sterowanie przechodzi do instrukcji następnej po instrukcji PERFORM; gdy <warunek> jest spełniony, wówczas wykonywany jest ciąg procedur od <nazwa-procedury-1> do <nazwa-procedury-2>, a potem sterowanie wraca do tej samej instrukcji PERFORM (czyli od nowa sprawdzana jest prawdziwość warunku itd.).

Instrukcja PERFORM zawarta w segmencie wymiennym może wywoływać jedynie procedury zawarte w tym samym segmencie wymiennym lub w segmencie stałym.

Instrukcja PERFORM zawarta w segmencie stałym może wywoływać procedury zawarte w segmencie stałym lub wymiennym.

Jeżeli w procedurach zawartych w segmencie wymiennym były instrukcje GO TO zmieniane instrukcją ALTER, to należy pamiętać, że w wyniku wywołania tego segmentu przez instrukcję PERFORM, procedury te będą w swojej pierwotnej postaci.

4.4.8.3.5. Instrukcja STOP

Instrukcja STOP powoduje chwilowe lub bezwzględne zatrzymanie działania programu.

Instrukcja STOP ma następującą postać ogólną:

$$\underline{\text{STOP}} \left\{ \begin{array}{l} \langle \text{tekst} \rangle \\ \underline{\text{RUN}} \end{array} \right\}$$

Przykłady

STOP RUN.

STOP 'NASTEPNY.SPWARIANT.LF'.

STOP 'USTAW.SPWSKAZNIK'.

Każdy program musi zawierać co najmniej jedną instrukcję STOP RUN, kończącą jego działanie. Po wykonaniu instrukcji STOP RUN sterowanie przekazywane jest systemowi operacyjnemu.

Wykonanie instrukcji STOP<tekst> powoduje wypisanie na monitorze tego tekstu i zawieszenie maszyny. Wystąpienie w tekście znaku linii (.LF) powoduje przerwanie wypisywania tekstu na monitorze.

Stan zawieszenia maszyny likwiduje się za pomocą dyrektywy START języka JOS (patrz ZAM 41 OPROGRAMOWANIE (SYSTEM 141), tom 1, JĘZYK OPERACYJNY SYSTEMU).

Po zlikwidowaniu stanu zawieszenia sterowanie programu przekazywane jest do instrukcji następnej za instrukcją STOP.

4.4.8.4. Instrukcje związane z sortowaniem

W języku COBOL dla maszyny ZAM 41 są trzy instrukcje związane z czynnością sortowania: SORT, RELEASE i RETURN.

4.4.8.4.1. Instrukcja SORT

Instrukcja SORT tworzy zbiór sortowania przez wykonanie procedur wejściowych lub pobranie dokumentów z innego zbioru. Następnie sortuje dokumenty zbioru sortowania wg zadanych kluczy.

Po zakończeniu sortowania udostępnia dokumenty zbioru sortowania procedurom wyjściowym lub zbiorowi wyjściowemu.

Postać ogólna instrukcji SORT podana jest we wkładce III na końcu rozdziału.

Przykłady

`SORT SORTOWANY ON DESCENDING KEY NAZWA..DOSTAWCY
USING WEJSCIE GIVING EWIDENCJA.`

`SORT SP ON ASCENDING SP1, SP2 INPUT PROCEDURE
IS ALFA THRU ALFA1 OUTPUT PROCEDURE IS WYJSCIE.`

<nazwa-zbioru-1> musi być nazwą zbioru sortowania opisaną ze wskaźnikiem poziomemu SD w DATA DIVISION.

Każda <nazwa-klucza> musi być nazwą pozycji elementarnej, nieindeksowanej, występującej we wszystkich dokumentach opisanych w zbiorze sortowania.

<nazwa-procedury-1> jest nazwą procedury wejściowej, a <nazwa-procedury-3> - nazwą procedury wyjściowej.

Zbiory: <nazwa-zbioru-2> oraz <nazwa-zbioru-3> muszą być zbiorami danych, tj. zbiorami opisanymi w DATA DIVISION ze wskaźnikiem poziomemu FD.

Część proceduralna może zawierać więcej niż jedną instrukcję SORT w dowolnym miejscu z wyjątkiem części DECLARATIVES, procedury wejściowej oraz procedury wyjściowej, związanych z instrukcją SORT.

Procedura wejściowa jest ciągiem jednej lub więcej procedur. Nie może ona być częścią ja-

kiejkolwiek procedury wyjściowej ani zawierać instrukcji SORT. Wewnątrz procedury wejściowej musi występować co najmniej jedna instrukcja RELEASE, której zadaniem jest przesyłanie dokumentów do zbioru sortowania.

Należy jednak pamiętać, że jej wykonanie jest sterowane instrukcją SORT. W związku z tym, gdy programista chce przekazać sterowanie do procedury wejściowej, nie w ramach instrukcji SORT, wówczas nie może on dopuścić do wykonania w tej procedurze instrukcji RELEASE. Poza tym ograniczeniem można procedurę wejściową traktować jak dowolny inny ciąg procedur w programie i przechodzić do jej wykonania z zewnątrz (a z procedury wejściowej - na zewnątrz) za pomocą instrukcji ALTER, GO TO oraz PERFORM.

Procedura wyjściowa jest ciągiem jednej lub kilku procedur; ciąg ten nie może być częścią żadnej procedury wejściowej ani zawierać instrukcji SORT. Wśród instrukcji procedury wyjściowej musi występować co najmniej jedna instrukcja RETURN służąca do pobierania dokumentów ze zbioru sortowania. Wykonanie instrukcji RETURN ma jednak sens tylko wtedy, gdy jest ono spowodowane przez instrukcje SORT. W związku z tym, gdy programista chce przekazać sterowanie do procedury wyjściowej nie w ramach instrukcji SORT, nie może on dopuścić do wykonania w tej procedurze instrukcji RETURN. Poza tym ograniczeniem procedurę wyjściową można traktować tak samo jak dowolny inny ciąg procedur w programie i realizować komunikację między nią a pozostałą częścią programu za pomocą instrukcji ALTER, GO TO oraz PERFORM.

Sposób sortowania zależy od zwrotów ASCENDING lub DESCENDING występujących w instrukcji SORT. Zależność tę określają następujące reguły:

- w przypadku użycia zwrotu ASCENDING zbiór porządkowany jest od najmniejszej wartości klucza do największej,
- w przypadku użycia zwrotu DESCENDING zbiór porządkowany jest od największej wartości klucza do najmniejszej.

Pozycje, które stanowią klucze muszą występować w opisie każdego dokumentu wymienionego w zwrocie DATA RECORDS zbioru sortowania. Następujące pravidła dotyczą kluczy sortowania:

- w każdym z dokumentów zbioru sortowania konkretny klucz musi być opisany tak samo i leżeć w tej samej odległości od początku dokumentu,
- w opisie klucza sortowania ani w opisie pozycji zawierających ten klucz nie może występować zwrot OCCURS,
- w jednej instrukcji SORT może występować najwyżej 10 kluczy sortowania.

Gdy w instrukcji SORT występuje zwrot INPUT PROCEDURE, wówczas sterowanie przekazywane jest do procedury wejściowej przed posortowaniem zbioru <nazwa-zbioru-1>. Translator COBOL wstawia na końcu procedury wejściowej specjalny mechanizm powrotu i po zakończeniu działania INPUT PROCEDURE następuje powrót do instrukcji SORT, a dokumenty włączone do zbioru sortowania za pomocą instrukcji RELEASE są wstępnie posortowane.

Jeżeli w instrukcji SORT występuje zwrot USING, to do zbioru sortowania <nazwa-zbioru-1> posyłane są automatycznie wszystkie dokumenty zbioru <nazwa-zbioru-2>. W chwili gdy zaczyna się wykonywanie instrukcji SORT zbiór <nazwa-zbioru-2> nie może być otwarty. Dla tego zbioru instrukcja SORT wykonuje automatycznie funkcje instrukcji OPEN INPUT, READ, CLOSE oraz komunikacji ze zdaniem USE.

Gdy w instrukcji SORT występuje zwrot OUTPUT PROCEDURE, wówczas po zakończeniu sortowania zbioru <nazwa-zbioru-1> sterowanie przekazywane jest do procedury wyjściowej. Translator COBOL wstawia na końcu procedury wyjściowej specjalny mechanizm powrotu, tak że w chwili gdy kończy się wykonywanie instrukcji SORT, wykonywanie programu jest kontynuowane od instrukcji następującej w programie po instrukcji SORT. Sterowanie przechodzi do procedury wyjściowej dopiero wtedy, gdy zbiór <nazwa-zbioru-1> jest już posortowany i można z niego pobierać kolejne dokumenty, uporządkowane według kluczy zadanych w instrukcji SORT. W procedurze wyjściowej kolejne dokumenty pobiera się ze zbioru sortowania za pomocą instrukcji RETURN.

Jeżeli w instrukcji SORT występuje zwrot GIVING, to posortowane dokumenty zbioru sortowania przesyłane są automatycznie do zbioru, którego nazwa wystąpiła w zwrocie GIVING. Zbiór ten w momencie rozpoczęcia wykonania instrukcji SORT nie może być otwarty. Funkcje instrukcji OPEN OUTPUT, WRITE i CLOSE oraz komunikacja z ewentualnym zdaniem USE wykonywane są automatycznie w ramach instrukcji SORT. Zbiorowi wystę-

pującemu w zwrocie GIVING można przypisać jedną z jednostek taśmowych zbioru sortowania. Używa się do tego zdania SELECT w wersji OR. Jeśli np. zbiorowi sortowania przydzielono jednostki: MTAPE A, MTAPE B i MTAPE D, to zbiorowi występującemu w zwrocie GIVING należy przydzielić MTAPE A, MTAPE B OR MTAPE D. Po zakończeniu sortowania zostanie wypisany komunikat zawiadamiający, na której jednostce znajduje się zbiór posortowany.

Zbiorowi występującemu w zwrocie USING można także przydzielić jedną z jednostek taśmowych zbioru sortowania z tym, że musi ona wystąpić jako ostatnia w zdaniu SELECT dla zbioru sortowania. Tej jednostki można używać także w procedurze wejściowej (INPUT PROCEDURE). Znajdująca się na tej jednostce szpula może być użyta jako szpula robocza sortowania, o ile pozwoli na to jej data ważności.

Jeżeli zbiór użyty w wersji USING nie jest zbiorem na taśmie magnetycznej i zawiera dokumenty różnej długości, to podczas sortowania krótsze dokumenty są dopełniane do maksymalnej długości przypadkowymi wartościami. Należy o tym pamiętać przy dalszym korzystaniu z posortowanych dokumentów.

Dla przyspieszenia czynności sortowania można zbiorowi sortowania w DATA DIVISION przydzielić większy bufor mieszczący kilka dokumentów (zwrotem BLOCK). Długość tego bloku nie może przekroczyć 2048 słów wraz z metryką (5 słów).¹⁾

1) zob. pkt 3.3.3.2.

4.4.8.4.2. Instrukcja RELEASE

Instrukcja RELEASE włącza dokument do zbioru sortowania.

Instrukcja RELEASE ma następującą postać ogólną:

RELEASE <nazwa-dokumentu >

Przykłady

RELEASE SORT..KARTA.

RELEASE S1.

Instrukcja RELEASE może być użyta tylko wewnątrz procedury wejściowej sortowania (INPUT PROCEDURE instrukcji SORT). W każdej procedurze wejściowej sortowania musi wystąpić co najmniej jedna instrukcja RELEASE. <nazwa-dokumentu> musi być nazwą dokumentu zbioru sortowania. Po zakończeniu procedury wejściowej zbiór sortowania zawiera tylko te dokumenty, które zostały do niego przesłane instrukcją RELEASE.

4.4.8.4.3. Instrukcja RETURN

Instrukcja RETURN przesyła kolejny dokument ze zbioru sortowania do pola dokumentu, gdzie jest dostępny dla programisty.

Instrukcja RETURN ma następującą postać ogólną:

RETURN<nazwa-zbioru> RECORD AT END GO TO
<nazwa-procedury>

Przykłady

RETURN ZBIOR..SORTOWANIA AT END GO TO S1.
RETURN SS1 AT END GO TO KONIEC..SORTOWANIA.

<nazwa-zbioru> użyta w instrukcji RETURN musi być nazwą zbioru sortowania, tj. zbioru opisanego w DATA DIVISION ze wskaźnikiem SD. Instrukcji RETURN wolno używać tylko w procedurach wyjściowych sortowania (OUTPUT PROCEDURE instrukcji SORT). W każdej procedurze wyjściowej sortowania musi wystąpić przynajmniej jedna instrukcja RETURN.

Instrukcja RETURN powoduje przesłanie kolejnego dokumentu z uporządkowanego wg zadanych kluczy zbioru sortowania do pola dokumentu związanego ze zbiorem sortowania. Jednocześnie do rejestru TALLY przesyłana jest długość tego dokumentu, liczona w słowach. Należy przy tym pamiętać, że dokumenty różnej długości ze zbiorów na nośnikach papierowych użytych w wersji USING (porównaj opis instrukcji SORT) zostały dopełnione do maksymalnej długości i ta długość będzie podana w rejestrze TALLY.

Wszystkie dokumenty przesyłane są oczywiście do jednego pola. Po wyczerpaniu wszystkich dokumentów zbioru sortowania sterowanie zostanie przekazane do procedury, której nazwa jest wymieniona we frazie AT END GO TO.

4.4.8.5. Instrukcje pomocnicze

W języku COBOL dla maszyny ZAM 41 istnieją dwie instrukcje pomocnicze: READY TRACE i RESET TRACE.

4.4.8.5.1. Instrukcja READY TRACE

Instrukcja READY TRACE służy do uruchamiania programu; od chwili wykonania instrukcji READY TRACE na wyjście określone w czołówce jako wyjście 0 programu¹⁾ wypisywane są po kolei nazwy wszystkich sekcji i paragrafów, przez które przechodzi sterowanie programu.

Instrukcja READY TRACE ma następującą postać ogólną:

READY TRACE

Dla każdej nazwy procedury wypisywane są dwa identyfikatory, oddzielone od siebie spacją. Pierwszy z nich jest nazwą sekcji, a drugi - nazwą paragrafu.

Wypisywanych jest co najwyżej po pięć znaków nazwy sekcji i nazwy paragrafu.

Jeżeli sterowanie przechodzi przez nazwę sekcji to zamiast nazwy paragrafu wypisywany jest symbol '***'.

Jeżeli sterowanie przechodzi przez nazwę paragrafu nie należącego do żadnej sekcji, to zamiast nazwy sekcji wypisywany jest symbol '***'.

Uwaga

Instrukcje READY TRACE są włączane do programu wynikowego tylko wtedy, gdy w czołówce programu źródłowego występuje dyrektywa TRA (porównaj pkt.5.3.1. Dyrektywy języka operacyjnego COBOL).

¹⁾ Wyjście 0 (zerowe) programu musi być zdefiniowane w czołówce systemowej programu z dekoderm z kodu wewnętrznego KW8.

4.4.8.5.2. Instrukcja RESET TRACE

Instrukcja RESET TRACE anuluje działanie instrukcji READY TRACE.

Instrukcja RESET TRACE ma następującą postać ogólną:

RESET TRACE

Od momentu wykonania instrukcji RESET TRACE ustaje wypisywanie nazw sekcji i paragrafów - spowodowane wykonaniem instrukcji READY TRACE.

4.4.9. JĘZYK WSTAWEK

Język wstawek został zdefiniowany jako podzbiór języka SAS (patrz ZAM 41, OPROGRAMOWANIE (SYSTEM 141), tom 2, JEZYK SAS). Umożliwia on pisanie programów, w których programista chce wykorzystać wszystkie właściwości maszyny, tj. działać na oddzielnych słowach maszyny przy użyciu wszystkich rozkazów maszyny użytkowej i z wykorzystaniem wszystkich jej rejestrów.

Poniżej podana jest składnia wszystkich elementów języka wstawek, podane są przykłady tych elementów i omówione ewentualne różnice między językiem wstawek a językiem SAS.

4.4.9.1. Podstawowe jednostki syntaktyczne

Część operacyjna rozkazu

Postać ogólna

$$\left\{ \begin{array}{l} \langle \text{litera} \rangle \langle \text{litera} \rangle \langle \text{litera} \rangle \\ \langle \text{litera} \rangle \langle \text{cyfra} \rangle \langle \text{cyfra} \rangle \end{array} \right\}$$

Przykłady

UMA

P39

Różnice

Części operacyjne są wyłącznie trzyznakowe. Wynika to z braku makrorozkazów w języku wstawek. Nie występują również rozkazy definiowane.

Etykieta

Postać ogólna

<identyfikator>

Przykład

DRUKOWANIE

WEJSCIE6

Uwaga

Identyfikator rozumiany jest tak jak w języku SAS a nie jak w języku COBOL (tj. w identyfikatorze nie może występować łącznik '..').

Tekst

Postać ogólna

$$\left[\left[\begin{array}{l} \langle \text{znak-podstawowy} \rangle \\ \langle \text{symbol-specjalny} \rangle \\ \langle \text{pochodna-znaku} \rangle \end{array} \right] \right]$$

Przykład

Tekst:

'TYTUL/::SP.QTSTILLER.QT'

zajmuje w maszynie 16 słów i znaczy

TYTUŁ: 'STILLER'

Różnice

Tekst rozumiany jest tak jak w języku COBOL. Należy zwrócić uwagę na to, że w odróżnieniu od języka SAS w języku wstawek każdy znak tekstu zajmuje w maszynie jedno słowo i zapisywany jest w kodzie KWS na ostatnich ośmiu bitach tego słowa.

Liczba

Postać ogólna

$$\left\{ \begin{array}{l} \text{+} \\ \text{-} \end{array} \right\} \left[\begin{array}{l} \text{[[<cyfra>]]} \cdot \text{[[<cyfra>]]} \\ \\ \text{. [[<cyfra>]]} \end{array} \right] \left[\begin{array}{l} \text{E} \left[\begin{array}{l} \text{+} \\ \text{-} \end{array} \right] \text{ [[<cyfra>]]} \end{array} \right]$$

Przykłady

-1234
-5121238744312
+6.28
+54E-28
-23.45E+2
-.5E-4

Różnice

Liczba rozumiana jest tak jak w języku COBOL.

Należy zwrócić uwagę na to, że - w odróżnieniu od języka SAS - w języku wstawek:

- a) liczby dwuprecyzyjne są wyłącznie całkowite,
- b) liczby ułamkowe są zapisywane w maszynie jako liczby zmiennoprzecinkowe,
- c) w zapisie liczby nie występują czynniki skalujące, zaczynające się w języku SAS od znaku '/'.

Szablon oktalny

Postać ogólna

$$= \left[\left\{ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 \right\} \right]$$

Przykład

$$\begin{aligned} &= 123 \\ &= 5000177 \\ &= 7777777710 \end{aligned}$$

Różnice

W języku wstawek wprowadzono dwa rodzaje szablonów:

szablon oktalny krótki,
szablon oktalny długi.

Szablon oktalny krótki jest ciągiem najwyżej ośmiu cyfr ze zbioru 0 1 2 3 4 5 6 7, zaczynającym się od znaku równości. W maszynie szablon oktalny krótki zapisywany jest w jednym słowie, przy czym wartość oktalna ostatniej cyfry szablonu zapisywana jest w ostatnich

trzech bitach tego słowa, wartość przedostatniej cyfry - w przedostatnich trzech bitach itd. Szablon oktalny krótki zawierający mniej niż osiem cyfr, jest równoważny szablonowi oktalnemu krótkiemu ośmiocyfrowemu, w którym pierwszymi cyframi są zera.

Szablon oktalny długi jest ciągiem zawierającym od dziewięciu do szesnastu cyfr ze zbioru 0 1 2 3 4 5 6 7, zaczynającym się od znaku równości. W maszynie szablon oktalny długi zapisywany jest w dwóch kolejnych słowach, przy czym wartość oktalna ostatniej cyfry szablonu zapisywana jest w ostatnich trzech bitach drugiego słowa, wartość przedostatniej cyfry - w przedostatniej trójce bitów tego słowa itd., wartość dziewiątej od końca cyfry - w ostatnich trzech bitach pierwszego słowa itd. Szablon oktalny długi zawierający więcej niż osiem, ale mniej niż szesnaście cyfr, jest równoważny szablonowi oktalnemu długiemu szesnastocyfrowemu, w którym pierwszymi cyframi są zera.

Parametr formalny

Postać ogólna

<identyfikator>

Przykład

X15

DRUGI

Różnice

W języku SAS nie występują parametry formalne. W trakcie translacji programu COBOL na miejsce parametrów formalnych we wstawkach posyła-

ne są adresy nazw danych przypisanych tym parametrom formalnym we frazie REPLACING zdania ENTER.

Wyrażenie adresowe

Postać ogólna

$$\left\{ \begin{array}{l} \langle \text{całk} \rangle [/ \langle \text{całk} \rangle] [\{ + | - \} \langle \text{całk} \rangle [[\langle \text{całk} \rangle]]] \\ * [\{ + | - \} \langle \text{całkowita} \rangle] \\ \langle \text{etykieta} \rangle [\{ + | - \} \langle \text{całkowita} \rangle] \\ \langle \text{parametr-formalny} \rangle [\{ + | - \} \langle \text{całkowita} \rangle] \end{array} \right\} \left\{ \begin{array}{l} [+] [*] \\ * + \end{array} \right\}$$

gdzie całk = całkowita

Przykład

1/11+3/17+7/23+1

* -12

START+8

Różnice

- W powyższym formacie $\langle \text{całkowita} \rangle$ oznacza liczbę całkowitą bez znaku,
- wyrażeniem adresowym nie może być stała (tj. liczba ułamkowa, liczba zmiennoprzecinkowa, tekst lub szablon), z wyjątkiem całkowitej bez znaku; w związku z tym dla wstawki nie jest tworzona lista stałych,
- wyrażenie adresowe postaci

$\text{całkowita} [/ \text{całkowita}] [[\{ + | - \} \text{całkowita} [/ \text{całkowita}]]]$

jest obliczane jako algebraiczna suma składników, a więc w tym samym wyrażeniu adre-

sowym mogą występować składniki dodatnie i ujemne; wynik tej sumy musi być mniejszy od 2^{15} ,

- d) adres symboliczny, tj. wyrażenie adresowe postaci

$\langle \text{etykieta} \rangle \{ + \mid - \} \langle \text{całkowita} \rangle$

oraz adres względny, tj. wyrażenie adresowe postaci

$* \{ + \mid - \} \langle \text{całkowita} \rangle$

nie może wskazywać punktu programu poza wstawką, w której on występuje, z wyjątkiem przypadku, gdy ten adres dotyczy pierwszego zdania COBOL po wstawce,

- e) za parametr formalny uważa się we wstawce każdy identyfikator nie będący etykietą zadeklarowaną w tej wstawce (por. paragraf: zdanie ENTER).
- f) w języku wstawek nie istnieje operacja pobrania adresu '='.

4.4.9.2. Opis języka

4.4.9.2.1. Rozkaz

Postać ogólna

$\langle \text{część-operacyjna-rozkazu} \rangle [\langle \text{wyrażenie-adresowe} \rangle]$

Przykład

UMA ALFA +
P31 16
SKO *-7
SDY 1/10+1/11+3/17+14

Różnice

Dotyczą tylko postaci wyrażenia adresowego i omówione są wyżej.

4.4.9.2.2. Dyrektywa REZE

Postać ogólna

REZE całkowita

Przykłady

REZE 16

REZE 100

Różnice

- a) Jedyną dyrektywą w języku wstawek jest REZE,
- b) w dyrektywie REZE nie może, tak jak w języku SAS - wystąpić liczba, która posyłana jest do wszystkich słów zarezerwowanych przez tę dyrektywę.

4.4.9.2.3. Komentarz

Postać ogólna

/*<ciąg-znaków-KW6-nie-zawierający-pary-
-znaków- ' * / ' > * /

Przykłady

/* OKRESLENIE ADRESU KLIENTA*/

/* DEFINIUJ PRZESLANIE*/

Uwaga

Między znakami / oraz * w ogranicznikach komentarza nie może wystąpić żaden znak.

4.4.9.2.4. Wiersz wstawki

Postać ogólna

$$\left[\left[\langle \text{etykieta} \rangle \right] \left\{ \begin{array}{l} \langle \text{tekst} \rangle \\ \langle \text{liczba} \rangle \\ \langle \text{szablon-oktalny} \rangle \\ \langle \text{rozkaz} \rangle \\ \langle \text{dyrektywa} \rangle \end{array} \right\} \left[\langle \text{komentarz} \rangle \right] \textcircled{\text{LF}} \textcircled{\text{CR}} \right]$$

Przykłady

'ZESTAWIENIE.SPWYNIKO/W'
-16.321E-18
=13701
DOA LB-3+ *
REZE 20

Różnica

W języku wstawek nie występują makrorozkazy, rozkazy definiowane oraz dyrektywy, z wyjątkiem dyrektywy REZE.

4.4.9.2.5. Wstawka

Postać ogólna

$\left[\left[\text{wiersz-wstawki} \right] \right]$

Przykład

UEB SWY+1

PMB 38

SDY 1/11+1/13+1/17+8 /*DOLACZ SYMBOLICZNE
WYJSCIE*/

UAD XY

UMB Z-5

PAD ILAT+10+

SOB * -1

UAD QY

PAD XY+2

UEB TX

PMB Z+2

UMB QY+2

PMB Z-2

UAD QY+3

PAD **2

SKO ILAT+40

+0

+0

QY) +0.874

=67

-10.9E-3

TX) STO 125

SWY) REZE 250 /*DEKODER*/

ILAT) REZE 40

Różnice

- a) Wstawka nie ma struktury blokowej,
- b) w języku wstawek nie występuje pojęcie sekcji.

Uwaga

- a) Identyfikatory zadeklarowane we wstawce znane są tylko w obrębie tej wstawki,
- b) zob. również paragraf Zdanie ENTER , w którym omówione są dalsze szczegóły związane z tworzeniem wstawek w programie napisanym w języku COBOL.

W K Ł A D K A III

1. Postać ogólna części proceduralnej

PROCEDURE DIVISION.

DECLARATIVES.

```

[[
  <nagłówek-sekcji>. <zdanie-USE>. {
    [[nagłówek-paragrafu. [[<zdanie>.]]]
    [[ <zdanie>.]]]
  }
]]

```

END DECLARATIVES.

```

{
  [[
    <nagłówek-sekcji>. {
      [[nagłówek-paragrafu. {
        <zdanie-COPY>. {
          [[<zdanie>.]]]
        }
      }
    ]]
  ]]
  [[
    <nagłówek-paragrafu>. {
      <zdanie-COPY>. {
        [[<zdanie>.]]]
      }
    }
  ]]
  [[ <zdanie>.] ]
}

```

END PROGRAM.

1.1. Postać ogólna nagłówka sekcji

<nazwa-sekcji> SECTION [<numer-segmentu>]

1.2. Postać ogólna nagłówka paragrafu

<nazwa-paragrafu>

1.3. Postać ogólna zdania

{
 <zdanie-ENTER>
 <zdanie-NOTE>
 <zdanie-EXIT>
 <zdanie-USE>
 <zdanie-COPY>
 <zdanie-END>
 [instrukcja]
 }

1.4. Postać ogólna instrukcji

<instrukcja-OPEN>|<instrukcja-READ>|<instrukcja-WRITE>|<instrukcja-CLOSE>|
 <instrukcja-ACCEPT>|<instrukcja-DISPLAY>|<instrukcja-COMPUTE>|<instrukcja-MOVE>|
 <instrukcja-EXAMINE>|<instrukcja-GO TO>|<instrukcja-IF>|<instrukcja-ALTER>|
 <instrukcja-PERFORM>|<instrukcja-STOP>|<instrukcja-SORT>|<instrukcja-RELEASE>|
 <instrukcja-RETURN>|<instrukcja-READY>|<instrukcja-RESET>

2. Postać ogólna zdania ENTER

ENTER SAS [REPLACING [BY <parametr-formalny> BY <nazwa-danej>]] ;

<wstawka>

ENTER COBOL

3. Postać ogólna zdania NOTE

NOTE <komentarz>

4. Postać ogólna zdania EXIT

<nagłówek-paragrafu>. EXIT

5. Postać ogólna zdania USE

WERSJA I

USE AFTER STANDARD ERROR PROCEDURE ON

INPUT
OUTPUT

WERSJA II

$$\text{USE} \left\{ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right\} \text{STANDARD} \left\{ \begin{array}{l} \text{BEGINNING} \\ \text{ENDING} \end{array} \right\} \left\{ \begin{array}{l} \text{REEL} \\ \text{FILE} \end{array} \right\} \text{LABEL PROCEDURE ON} \left\{ \begin{array}{l} \text{INPUT} \\ \text{OUTPUT} \end{array} \right\} \langle \text{nazwa-zbioru} \rangle$$

6. Postać ogólna zdania COPY

COPY $\langle \text{nazwa-biblioteczna} \rangle$

7. Postać ogólna zdania END

$$\text{END} \left\{ \begin{array}{l} \text{DECLARATIVES} \\ \text{PROGRAM} \end{array} \right\}$$

8. Postać ogólna instrukcji OPEN

$$\text{OPEN} \left\{ \begin{array}{l} \text{INPUT} \\ \text{OUTPUT} \end{array} \right\} \left[\begin{array}{l} \text{ } \\ \text{ } \end{array} \right] \langle \text{nazwa-zbioru} \rangle \left[\begin{array}{l} \text{ } \\ \text{ } \end{array} \right] \left\{ \begin{array}{l} \text{INPUT} \\ \text{OUTPUT} \end{array} \right\} \left[\begin{array}{l} \text{ } \\ \text{ } \end{array} \right] \langle \text{nazwa-zbioru} \rangle \left[\begin{array}{l} \text{ } \\ \text{ } \end{array} \right]$$

9. Postać ogólna instrukcji READ
READ <nazwa-zbioru> RECORD AT END GO TO <nazwa-procedury>
10. Postać ogólna instrukcji WRITE
WRITE <nazwa-dokumentu>
11. Postać ogólna instrukcji CLOSE
CLOSE [, <nazwa-zbioru> [REEL] [WITH { LOCK } { NO REWIND }]]]
12. Postać ogólna instrukcji ACCEPT
ACCEPT [, <nazwa-jednostki-danych>] [FROM INPUT <całkowita>]]
13. Postać ogólna instrukcji DISPLAY
DISPLAY [, [(<wyrażenie-arytmetyczne>)]] { <tekst> <nazwa-jednostki-danych> HIGH..VALUE LOW..VALUE SPACE BUFF } }

14. Postać ogólna instrukcji COMPUTE

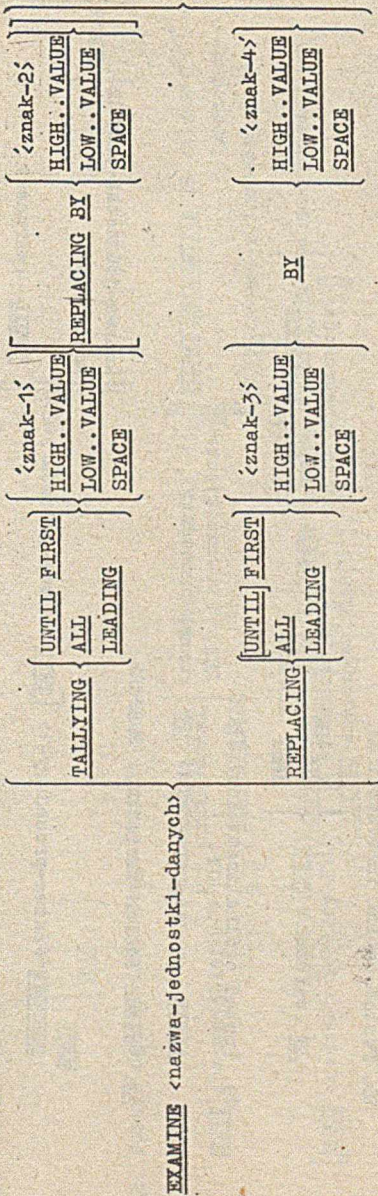
$$\left\{ \begin{array}{l} \underline{\text{COMPUTE}} \\ \underline{\text{COMP}} \end{array} \right\} \langle \text{nazwa-jednostki-danych} \rangle \left[\underline{\text{ROUNDED}} \right] = \langle \text{wyrażenie-arytmetyczne} \rangle$$

$$\left[\underline{\text{ON SIZE ERROR GO TO}} \langle \text{nazwa-procedury} \rangle \right]$$

15. Postać ogólna instrukcji MOVE

$$\left\{ \begin{array}{l} \langle \text{nazwa-jednostki-danych} \rangle \\ \langle \text{stała} \rangle \\ \underline{\text{HIGH..VALUE}} \\ \underline{\text{LOW..VALUE}} \\ \underline{\text{ZERO}} \\ \underline{\text{SPACE}} \\ \underline{\text{TALLY}} \\ \underline{\text{SECT}} \\ \underline{\text{IDENT}} \\ \underline{\text{CREAT}} \\ \underline{\text{GENER}} \\ \underline{\text{EXPIR}} \end{array} \right\} \left\{ \begin{array}{l} \underline{\text{MOVE}} \\ \underline{\text{TO}} \left[\langle \text{nazwa-jednostki-danych} \rangle \right] \end{array} \right\}$$

16. Postać ogólna instrukcji EXAMINE



17. Postać ogólna instrukcji GO TO

GO TO <nazwa-procedury> [[; <nazwa-procedury>]] DEPENDING ON <nazwa-jednostki-danych>

18. Postać ogólna instrukcji IF

IF <warunek> THEN { [; <instrukcja>]] ELSE { [; <instrukcja>]] }
NEXT SENTENCE NEXT SENTENCE

19. Postać ogólna instrukcji ALTER

ALTER <nazwa-paragrafu> TO <nazwa-procedury>

20. Postać ogólna instrukcji PERFORM

PERFORM <nazwa-procedury-1> [THRU <nazwa-procedury-2>] { { <nazwa-jednostki-danych> } TIMES }
UNTIL <warunek>

21. Postać ogólna instrukcji STOP

STOP {<tekst> RUN}

22. Postać ogólna instrukcji SORT

$$\left\{ \left[\left[\text{SORT } \langle \text{nazwa-zbloru-1} \rangle \text{ ON } \left\{ \begin{array}{l} \text{DESCENDING} \\ \text{ASCENDING} \end{array} \right\} \text{ KEY } \left[\begin{array}{l} \text{,} \\ \text{,} \end{array} \right] \langle \text{nazwa-klucza} \rangle \right] \right] \left\{ \begin{array}{l} \text{INPUT PROCEDURE IS } \langle \text{nazwa-} \\ \text{procedure-1} \rangle \\ \text{USING } \langle \text{nazwa-zbloru-2} \rangle \end{array} \right\} \right\}$$

• 23. Postać ogólna instrukcji RELEASE

RELEASE
< nazwa - dokumentu >

24. Postać ogólna instrukcji RETURN

RETURN <nazwa-zbioru> RECORD AT END GO TO <nazwa-procedury>

25. Postać ogólna instrukcji READY TRACE

READY TRACE

26. Postać ogólna instrukcji RESET TRACE

RESET TRACE

5. WIADOMOŚCI UZUPEŁNIAJĄCE

5.1. ZBIORY BIBLIOTECZNE

5.1.1. KORZYSTANIE Z BIBLIOTEKI COBOL

Biblioteka COBOL jest częścią Biblioteki Systemu SO 141. Biblioteka COBOL składa się ze zbiorów, z których każdy zawiera pewien fragment programu źródłowego COBOL. Fragmenty te można włączać - za pomocą zdań COPY - do dowolnego programu źródłowego.

Za pomocą jednego zdania COPY włącza się jeden określony fragment. Włączanie to odbywa się we wstępnym etapie translacji i polega na zastąpieniu zdania COPY<nazwa-biblioteczna> zawartością zbioru bibliotecznego o wymienionej nazwie.

Reguły dotyczące korzystania ze zbiorów bibliotecznych zostały podane przy opisie zdania COPY w pkt 4.4.

5.1.2. TWORZENIE ZBIORÓW BIBLIOTECZNYCH ¹⁾

W celu włączenia zbioru do Biblioteki należy wyprodukować jego taśmę binarną. Włączenie taśmy binarnej do Biblioteki Systemu wykonuje się według ogólnych zasad (za pomocą języka BIB2). Taśmę binarną produkuje się za pomocą języków COP1 i BIB1. Dołączany zbiór zapisujemy w COBOL-u. Znakiem końca zbioru bibliotecznego jest dwukropek.

Aby wyprodukować taśmę binarną zbioru bibliotecznego należy wprowadzić do maszyny następujące informacje:

- a) czołówkę w JOM definiującą symboliczne wejście-wyjście i problem,
- b) tekst zbioru bibliotecznego zakończony dwukropkiem - z symbolicznego wejścia 0,
- c) program w języku JB1 - z symbolicznego wejścia 1,
- d) program w języku PJEG - z symbolicznego wejścia 0.

Ad a) Czołówka jest redagowana według ogólnych zasad z tym, że zdanie PROBLEM musi mieć postać: PROBLEM : COP1, BIB1.

Język COP1 wczytuje zbiór z wejścia 0 i przetwarza do postaci wymaganej

1) zob. MASZYNA ZAM 41, SYSTEM OPROGRAMOWANIA SO 141, OPISY FUNKCJONALNE (materiały aktualizowane).

przez translator COBOL-u. Ewentualny błąd sygnalizowany jest na wyjściu 0. Przygotowany przez COP1 zbiór wyprowadzany jest następnie w postaci binarnej przez BIB1. BIB1 korzysta z następujących symbolicznych WE/WY:

WEJSCIE 1 - program w języku JB1,
WYJSCIE 0 - sygnalizacja błędów,
WYJSCIE 1 - komentarz,
WYJSCIE 2 - taśma binarna.

Wszystkie te symboliczne wejścia-wyjścia, a także symboliczne wejścia-wyjścia COP1 i PJEG powinny być zdefiniowane w czołówce jako wejścia i wyjścia translatora.

Ad b) Tekst zbioru bibliotecznego stanowi dane do COP1. Gdy przetworzony zbiór nie mieści się w 2048 słowach maszyny, wówczas tekstem - ZA DŁUGI ZBIOR DO NAGRA-
NIA sygnalizowany jest błąd.

Ad c) Program w języku JB1:

Z : NZB = XXXX,
ORG = Y,
TYP = TEKST,
PZS = XXXX;
P : NPZ = XXXX;
T : TRA = PJEG;
KONIEC;

Uwaga

Zamiast XXXX należy wstawić nazwę nagrywanego zbioru bibliotecznego. Nazwa nie może pokrywać się z nazwą zbioru już występującego w Bibliotece Systemu.

Ad d) Program w języku PJEG jest ustalony i ma następującą postać:

```
VPB[ ]
VCM[6001]
VRZ[K, (1)]
BLAD
:1
VMB[16000]
VWP[ 4000]
VCM[ 6000]
VDP[ K]
VPP[ 3000]
VCM[ 6002]
VKB
VST[ 0,0,K,K]
```

Przykład

```
WE : 1 = PCTP (CTP1, IM2C);
WY : 0 = PDW (DW, DDW1),
      1 = PPTP(PTP1, IM2P),
      2 = PPTP(PTP1, DJED);
TYTUL : TASMA BINARNA;
PROBLEM : COP1, BIB1.
<odstęp na taśmie>
```

```
. 02 KOLES.
03 IMIE ; PICTURE IS 'A(10)'.
03 NAZWISKO ; PICTURE IS 'A(20)'.
03 PLEC ; PICTURE IS '9'.
02 DATA1.
03 DZIEN ; PICTURE IS '99'.
03 MIESIAC ; PICTURE IS '99'.
:
```

<odstęp na taśmie>

```
Z : NZB = CQCQ,
ORG = Y,
TYP = TEKST,
PZS = CQCQ;
```

```
P:NPZ = CQCQ ;  
T : TRA = PJEG ;  
<odstęp na taśmie>  
VPB [ ]  
VCM [6001]  
VRZ [K, (1)]  
BLAD  
: 1  
VWB [16000]  
VWP [ 4000]  
VCM [ 6000]  
VDP [K]  
VPP [3000]  
VCM [6002]  
VKB  
VST [0,0,K,K]  
<odstęp na taśmie>  
KONIEC ;
```

Wczytując wyperforowaną zgodnie z powyższym wzorem taśmę z czytnika 1 otrzymamy w wyniku na PTP1 taśmę binarną zbioru o nazwie CQCQ zawierającego opis pozycji KOLES.

Po włączeniu zbioru CQCQ do Biblioteki można będzie np. napisać:

```
01 PRACOWNIK  
COPY CQCQ.
```

5.2. RERUN - PONAWIANIE PROGRAMU

Ponawianie programu polega na wznowianiu jego wykonywania od stanu w jakim wykonywanie to zostało przerwane.

Co pewien czas zapamiętywany jest stan całej maszyny, to znaczy zawartość pamięci operacyjnej i bębnowej oraz stan wszystkich urządzeń zewnętrznych i rejestrów wewnętrznych. Jest to tak zwana operacja RERUN. W którym momencie

wykonywana jest ta operacja określa zwrot RERUN w części ENVIRONMENT, mianowicie:

- po przesłaniu określonej liczby dokumentów zbioru danych lub sortowania,
- po napotkaniu końca szpuli w danych związanych z taśmą magnetyczną,
- po napotkaniu w danych wejściowych pewnego dokumentu (specjalnie do tego celu przeznaczonego) zwanego oznaczniakiem operacji ponawiania.

Stan maszyny zapamiętywany jest na taśmie magnetycznej w postaci tzw. bloków rerunowych. Może to być taśma specjalnie do tego celu przeznaczona lub jedna z taśm aktualnie opracowywanego zbioru danych.

W zwrocie RERUN wymienia się nazwę zbioru danych, w czasie opracowywania którego wykonuje się operację RERUN, a także nazwę zbioru związanego z taśmą magnetyczną, na której mają znaleźć się bloki rerunowe.

Oznacznik operacji ponawiania jest dokumentem specjalnym zbudowanym z bezpośrednio po sobie następujących czterech znaków, zwanych znakami rerunowymi oraz ogranicznika dokumentu. Dokument ten może się pojawić tylko w danych wejściowych związanych z nośnikiem papierowym. Znak rerunowy określa się w zwrocie APPLY w części ENVIRONMENT.

Po wykonaniu operacji RERUN na monitor wypisywany jest tekst:

```
WYKONANO RERUN <identyfikator-zbioru>  
                <symbol-urządzenia>  
                <numer-bloku>
```

gdzie:

- <identyfikator-zbioru> jest identyfikatorem zbioru, w którym zostały umieszczone bloki rerunowe,
- <symbol-urządzenia> jest symbolem stacji, na której dokonano zapisu bloków rerunowych; <symbolem stacji> jest jedna z liter A, B, F
- <numer-bloku> jest numerem bloku pierwszego bloku rerunowego.

W trakcie wczytywania dokumentów ze zbioru danych, związanego z nośnikiem papierowym, liczone są wczytane dokumenty. Służy do tego licznik dokumentów. Wykorzystywane jest to do ustawiania urządzeń wejściowych w tym samym położeniu, w jakim znajdowały się w chwili wykonania operacji RERUN. Aby przy ponawianiu programu uniknąć konieczności ustawiania zbioru danych zawsze od początku - wprowadzono specjalny dokument zwany zerowatorem licznika dokumentu.

W momencie natrafienia w danych na zerowator, numer zerowatora zostaje zapamiętany, a licznik dokumentów wyzerowany. Przy wznowianiu programu taśma z danymi może być założona gdziekolwiek przed zerowatorem, który został wczytany jako ostatni przed ostatnio wykonaną operacją RERUN. Ustawianie danych na właściwym miejscu odbywa się już w trakcie akcji ponawiania programu automatycznie.

Zerowator licznika dokumentów ma postać bezpośrednio po sobie następujących trzech znaków końca zbioru, liczby całkowitej oraz ogranicznika dokumentu. Liczba całkowita jest numerem zerowatora.

Wznowienie wykonania programu przeprowadza się za pomocą translatora RESTART, poza językiem COBOL. Aby dokonać tej operacji należy zbudować czołówkę wg następujących reguł:

- 1) zdanie "zdefiniuj problem" czołówki problemu musi mieć postać

PROBLEM: RESTART, PROGRAM.

- 2) po czołówce problemu następuje czołówka translatora RESTART o następującej budowie:

U = <symbol-urządzenia>, B = <numer-bloku>

<symbol-urządzenia> jest tu literą określającą stację taśmy magnetycznej, na której jest szpula zawierająca bloki rerunowe,

<numer-bloku> jest liczbą całkowitą, która jest numerem pierwszego bloku rerunowego. Numer ten jest podawany w komunikacie wypisywanym po wykonaniu operacji RERUN.

Ponadto szpulę zawierającą bloki rerunowe danego programu należy założyć na stacji taśmy magnetycznej, której symbol określony jest w czołówce translatora RESTART.

Po wczytaniu czołówki translatora RESTART na monitorze zostają wypisane komunikaty określające nośniki i urządzenia, z którymi związane są poszczególne zbiory danych opracowywane przez wznowiany program.

Komunikaty te mają następującą postać:

• dla zbiorów na taśmach magnetycznych -

ZBIOR GENERACJA SZPULA <identyfikator-zbioru>

<numer-generacji> <numer-szpuli>

ZALÓŻ NA <symbol-stacji>

- dla zbiorów związanych z nośnikiem papierowym -

ZBIOR ZEROWATOR <identyfikator zbioru>
<numer zerowatora>

ZALOŻ NA <numer-symbolicznego-wejścia>

Po wypisaniu tych komunikatów maszyna zostaje zawieszona. W czasie zawieszenia maszyny - nośniki zawierające zbiory danych należy założyć na odpowiednie urządzenia zewnętrzne.

Nośniki każdego ze zbiorów związanych z symbolicznym wejściem-wyjściem należy zakładać przed zerowatorem <numer zerowatora>. Gdy <numer-zerowatora> jest równy zeru, wówczas nośnik należy założyć od początku. Ponieważ od tego momentu taśma magnetyczna z blokami rerunowymi nie jest już potrzebna, można ją zdjąć, a tym samym zwolnić zajęta przez nią stację.

Po umieszczeniu wszystkich zbiorów danych na odpowiadających im urządzeniach należy odwieść maszynę. Wykonanie programu będzie kontynuowane.

5.3. JĘZYK OPERACYJNY

5.3.1. DYREKTYWY JĘZYKA OPERACYJNEGO COBOL-U

Program w języku COBOL może być poprzedzony czołówką będącą ciągiem dyrektyw języka operacyjnego COBOL-u. Język operacyjny COBOL-u składa się z siedmiu dyrektyw; przed danym programem mogą wystąpić wszystkie dyrektywy lub tylko niektóre, zależnie od potrzeb.

Poszczególne dyrektywy muszą być oddzielone średnikami, po ostatniej dyrektywie musi wystąpić kropka. Kolejność występowania dyrektyw w czołówce jest nieistotna.

5.3.1.1. Dyrektywa PAM

Postać ogólna

PAM

Funkcja

Dyrektywa PAM powoduje wypisanie na wyjściu symbolicznym 0 translatora parametrów określających rozmieszczenie programu wynikowego w pamięci operacyjnej i na bębnie.

Parametry programu wypisywane są w następującej postaci:

	APAO	MIEJSC	ABEB	MIEJSC
DANE	adres	całkowita		
SEGMENTY				
0	adres	całkowita	adres	
1	adres	całkowita	adres	
:	:	:	:	
WOLNE	adres	całkowita	adres	całkowita

Przez DANE rozumie się tu obszar zarezerwowany na pola wszystkich zbiorów i dokumentów.

Długość segmentu na bębnie jest taka sama jak długość segmentu w PAO.

5.3.1.2. Dyrektywa SLA

Postać ogólna

SLA

Funkcja

Dyrektywa SLA powoduje wypisanie na zerowym wyjściu programu - po zakończeniu wykonywania programu - nazwy ostatnio wykonywanej sekcji i paragrafu według reguł podanych w opisie instrukcji READY TRACE (patrz pkt 4.4.8.5.1.)

5.3.1.3. Dyrektywa BIB

Postać ogólna

BIB([nazwa-zbioru])

Funkcja

Dyrektywa BIB powoduje sprowadzenie z taśmy systemu tych zbiorów bibliotecznych, których nazwy są w niej wymienione.

Dyrektywa ta musi wystąpić przed każdym programem zawierającym choć jedno zdanie COPY, mianowicie argument każdego zdania COPY w programie musi równocześnie występować na liście dyrektywy BIB poprzedzającej ten program.

Sposób dołączania do Biblioteki Systemu fragmentów programu COBOL w postaci zbioru opisano w pkt 5.1.

5.3.1.4. Dyrektywa TRA

Postać ogólna

TRA

Funkcja

Dyrektywa ta powoduje, że występujące w programie źródłowym instrukcje READY TRACE są włączane do programu wynikowego.

Jeśli przed danym programem nie występuje dyrektywa TRA, to występujące w nim instrukcje READY TRACE są traktowane jak komentarz; umożliwia to nieusuwanie tych instrukcji z programu po jego uruchomieniu.

5.3.1.5. Dyrektywa TAB

Postać ogólna

TAB

Funkcja

Dyrektywa TAB powoduje wypisanie - na zerowe wyjście symboliczne translatora - tabulogramu programu źródłowego i czołówki COBOL. Tabulogram wypisywany jest w trakcie translacji.

- a) Do tabulogramu nie wchodzi komentarze z PROCEDURE DIVISION ani tekst komentarzowy, napisany po nazwie programu w paragrafie PROGRAM...ID części IDENTIFICATION DIVISION.
- b) Obok zdań COPY występujących w programie źródłowym, w tabulogramie umieszczane są zawartości sprowadzonych zbiorów bibliotecznych.

5.3.1.6. Dyrektywa MON

Postać ogólna

MON

Funkcja

Dyrektywy MON używa się w celu zaznaczenia, że w trakcie wykonywania programu mają być wypisywane monity sygnalizujące każde otwarcie i zamknięcie zbioru lub szpuli, mianowicie:

następujące trzy monity (patrz Dodatek H) wypisywane są tylko wtedy, gdy program źródłowy poprzedzony jest dyrektywą MON:

OTWARTO ZBIOR<identyfikator-zbioru>symbol-
urządzenia>
ZAMKNIETO ZBIOR<identyfikator-zbioru>symbol-
urządzenia>
ZAMKNIETO ZBIOR LUB SZPULE<identyfikator-zbio-
ru>symbol-urządzenia>
<ilość-błędnych-przesłań>

Pozostałe monity nie są związane z dyrektywą MON.

5.3.1.7. Dyrektywa IGN

Postać ogólna

IGN

Funkcja

Program, który jest poprzedzony dyrektywą IGN może być wykonywany nawet wtedy, gdy wystąpią w nim błędy, jeśli tylko zostanie przetłumaczony do końca, czyli jeśli będą spełnione obydwa poniższe warunki:

- a) żaden z błędów nie należał do grupy błędów przerywających tłumaczenie,

b) liczba błędów nie przekracza 32.

5.3.2. MONITY I SYGNALIZACJA BŁĘDÓW

Translator COPOL wykrywa występujące w programie źródłowym błędy czyli odstępstwa translowanego programu od postaci prawidłowej opisanej w tym podręczniku.

Błędy dzielą się na dwie grupy:

- a) błędy wykrywane w czasie translowania programu, czyli tzw. błędy czasu tłumaczenia,
- b) błędy wykrywane dopiero w czasie wykonania programu czyli tzw. błędy czasu wykonania.

Błędy czasu tłumaczenia są sygnalizowane na zerowym wyjściu symbolicznym translatora. Sygnalizacja błędu polega na wypisaniu odpowiedniego tekstu określającego rodzaj błędu (spis wszystkich rodzajów monitów czasu tłumaczenia - zob. Dodatek E) oraz tzw. "współrzędnych błędu" określających miejsce, w którym wystąpił błąd.

Na ogół po zasygnalizowaniu błędu tłumaczenie programu jest kontynuowane; istnieją jednak takie błędy (np. odwołanie do niewystępującego w Bibliotece zbioru bibliotecznego), po stwierdzeniu których tłumaczenie jest przerywane.

Należy tu zwrócić uwagę na to, że czasem jeden błąd w programie może spowodować sygnalizację więcej niż jednego tekstu błędu; te niekorzystne przypadki wynikają z budowy translatora.

Jeśli w czasie tłumaczenia programu zostanie stwierdzony chociaż jeden błąd, to niezależnie od monitów o rodzaju poszczególnych błędów - translator języka COBOL na zakończenie swej pracy wypisuje monit "BYLY BLEDY".

Analogiczna jest sytuacja w przypadku błędów czasu wykonania: przy niektórych błędach działanie programu jest przerywane (np. przy próbie wykonania otwarcia zbioru już otwartego); w innych przypadkach praca programu jest zawieszana w celu umożliwienia operatorowi wykonania odpowiedniej akcji (np. podłożenie prawidłowej szpuli).

Błędy czasu wykonania są sygnalizowane na zerowym wyjściu symbolicznym programu. Spis wszystkich monitów czasu wykonania podany jest w Dodatku F; w Dodatku tym jest też omówione znaczenie poszczególnych monitów.

Oprócz sygnalizacji błędów istnieją także monity sygnalizujące operatorowi albo stan realizacji programu (np. "ZAMKNIETO ZBIOR<identyfikator-zbioru>xsymbol-urządzenia>") albo konieczność wykonania pewnych czynności (np. "ZMIEN SZPULE<identyfikator-zbioru>xsymbol-urządzenia>").

5.3.3. TRANSLATOR ZAKOŃCZEŃ

Jeśli w czołówce problemu, w zdaniu PROBLEM, po instrukcji PROGRAM wystąpi instrukcja ZCOB, to po zakończeniu działania programu będzie wywołany Translator Zakończeń.

Translator Zakończeń sygnalizuje na zerowym

wyjściu symbolicznym programu monit interpretujący przyczynę zakończenia programu.

Alfabetyczny spis monitów Translatora Zakończeń podany jest w Dodatku G.

5.3.4. CZOŁÓWKA WYMAGANA PRZEZ SYSTEM OPERACYJNY SO 141

Program źródłowy musi być poprzedzony czołówką wymaganą przez System Operacyjny SO 141 (patrz ZAM 41, OPROGRAMOWANIE (SYSTEM 141), Tom I, MASZYNA UŻYTKOWA).

W czołówce problemu translator języka COBOL wywoływany jest za pomocą nazwy COBO lub COBOL; natomiast cobolowski Translator Zakończeń wywoływany jest za pomocą nazwy ZCOB,np. PROBLEM: COBO, PROG, ZCOB.

Symboliczne wejścia/wyjścia używane w programie muszą być zdefiniowane w czołówce problemu. Symboliczne wejścia/wyjścia w programie COBOL mogą być używane jako:

- a) wejścia/wyjścia przydzielone zbiorom przetwarzanym w programie (zob. deklaracja SELECT w części ENVIRONMENT),
- b) wejścia, z których dane są wczytywane za pomocą instrukcji ACCEPT lub wyjścia, na które informacje są wypisywane za pomocą instrukcji DISPLAY,
- c) wyjście związane z wykonaniem instrukcji READY TRACE lub dyrektywy SLA (są to zawsze wyjścia zerowe).

W przypadku użycia instrukcji READY TRACE i dyrektywy TRA zerowe wyjście programu musi

być zdefiniowane w czołówce problemu za pomocą dekodera z kodu wewnętrznego KW8. Zerowe wejście programu i zerowe wyjście programu jest dołączane automatycznie (jeśli jest to czytnik i perforator taśmy papierowej, to nie trzeba ich nawet definiować w czołówce). Natomiast dla symbolicznych wejść/wyjść programu o numerach różnych od zera programista sam musi:

- a) zdefiniować w czołówce dekodery i urządzenie przypisując mu numer 1-15,
- b) zarezerwować bufor na dekodery za pomocą języka wstawek SDY DOLACZ - (patrz ZAM 41, OPROGRAMOWANIE SYSTEM 141, Tom I, MASZYNA UŻYTKOWA).

Wejścia przydzielone danym wejściowym muszą być definiowane za pomocą dekoderek z kodu, w którym dane są zapisane na nośniku zewnętrznym na kod KW8, a wyjścia przydzielane danym wyjściowym muszą być zdefiniowane za pomocą dekoderek z kodu KW8 na kod, w którym chcemy otrzymać wynik.

Chcemy w tym miejscu zwrócić uwagę na dość częsty błąd popełniany przy pisaniu programów przetwarzających zbiór związany z czytnikiem kart, a mianowicie: ponieważ w COBOL-u instrukcje READ, WRITE, ACCEPT i DISPLAY są wykonywane za pomocą rozkazu P31, to używać można tylko dekoderek podających po jednym znaku, a więc nie można używać dekodera PCK, lecz tylko PCKP.

5.3.5. PRZYKŁAD UŻYCIA SYMBOLICZNEGO WEJŚCIA I WYJŚCIA

Wyobraźmy sobie, że w programie chcemy przetwarzać zbiór wejściowy związany z taśmą papierową i otrzymywać zbiór wyjściowy związany także z taśmą papierową. Zbiory te nazwiemy w programie odpowiednio ZBIOR..WE i ZBIOR..WY; dokumenty tych zbiorów będziemy chcieli dziurkować (otrzymywać) w kodzie M2.

Niech sytuacja będzie taka, że zerowe wyjście programu jest związane z drukarką wierszową - gdyż w programie użyta jest instrukcja READY TRACE, której wyniki najwygodniej jest otrzymać właśnie na drukarce. Wówczas perforator na dziurkowanie dokumentów zbioru ZBIOR..WY trzeba związać z symbolicznym wyjściem o numerze np. 1, dokumenty zbioru ZBIOR..WE będziemy chcieli czytać za pomocą czytnika 1 związanego z symbolicznym wejściem o numerze 1.

W czołówce tego programu symboliczne wejścia trzeba zdefiniować następująco:

```
WYP:0 = PDW(DW,DDV8);  
WYP:1 = PPTP(PTP1, D8M2);  
WEP:1 = PCTP(CTP1, DM28);
```

Zerowe wyjście jest dołączane automatycznie, natomiast wejście o numerze 1 i wyjście o numerze 1 trzeba dołączyć za pomocą następującej wstawki SAS:

```
ENTER SAS;  
SKO A  
B) REZE 480
```

A) UEB B

PMB 38

SDY 1/11+1/13+1/17+8

SDY 1/11+1/17+8

ENTER COBOL.

Dołączenie symbolicznych wejść/wyjść obowiązuje albo w całym programie COBOL, jeśli wstawka SAS została umieszczona w segmencie stałym, albo tylko w tym segmencie wymiennym, w którym została umieszczona wstawka.

5.4. UWAGI KOŃCOWE

Rozdział ten zawiera pewne uwagi i praktyczne wskazówki dla programujących w języku COBOL. Niektóre z poruszonych tu zagadnień były już omówione w poprzednich częściach niniejszego opracowania. Zebrane tu informacje powinny być pomocne szczególnie dla mniej doświadczonych programistów.

5.4.1. SIECI DZIAŁAŃ

Wielu początkujących programistów uważa, że napisanie programu w języku COBOL nie wymaga wcześniejszego przygotowania sieci działań dla części proceduralnej.

Jak wynika z doświadczeń programowania w języku COBOL, w wielu programach pisanych bez sieci działań, zdarzają się błędy w logicznej strukturze programu, duże fragmenty niemal identycznych procedur wypisywane są wielokrotnie, co wydłuża niepotrzebnie program i powoduje zbyteczne zajmowanie miejsc pamięci itp.

Bez sieci działań można pisać jedynie programy krótkie i proste. Jeżeli jednak program będzie długi lub logicznie skomplikowany, jest rzeczą konieczną przygotowanie przed napisaniem programu sieci działań. Istnieje kilka różnych, a zalecanych przez podręczniki programowania, notacji sieci działań. Z punktu widzenia przydatności do pisania programu i sporządzania dokumentacji, jest rzeczą obojętną jaką notację przyjmie programista. Wobec tego pominiemy tu szczegółowy opis notacji, ograniczając się do załączenia sieci działań programu przykładowego w Dodatku H.

5.4.2. OPTYMALNE PISANIE PROGRAMÓW

Maszyna ZAM 41, na której zrealizowano opisaną wersję języka COBOL, należy do rzędu maszyn średnich zarówno pod względem pojemności pamięci jak i szybkości działania. W związku z tym jest rzeczą bardzo ważną, aby programy w języku COBOL dla maszyny ZAM 41 pisane były możliwie optymalnie, ze względu na dostępny obszar pamięci wewnętrznej i bębnowej oraz czas pracy maszyny. Należy jednak zwrócić uwagę na fakt, że często optymalizacja programu z punktu widzenia zajmowanego miejsca może powodować przedłużenie czasu jego działania i odwrotnie. Programista musi wtedy zdecydować która wersja jest korzystniejsza lub przyjąć wyjście kompromisowe.

Często spotykanym przykładem "rozrzutności" programisty jest tworzenie nadmiernej liczby rodzajów dokumentów w opracowywanych zbiorach.

Można tego uniknąć przez umiejętne stosowanie znaczników dokumentów lub nazw warunkowych i tworzenie dokumentów możliwie uniwersalnych. Wykorzystanie wspólnych pól (opcja SAME RECORD AREA) dla dokumentów różnych zbiorów, prócz oszczędności miejsca przeznaczonego na te pola, w wielu przypadkach pozwala pominąć instrukcję MOVE przy przepisywaniu i aktualizacji zbiorów. Instrukcje ALTER i PERFORM dają także duże możliwości optymalizacji programów.

Osobnym zagadnieniem jest umieszczanie tekstów w programie, gdyż zajmują one dużo miejsca (zapisywane są po jednym znaku w słowie maszyny). Czasem warto więc umieszczać duże teksty w zbiorach wejściowych oraz korzystać z replikatora w instrukcji DISPLAY przy konieczności wypisywania wielu jednakowych znaków przy podkreśleniach, tabelkach, wykresach itp., np. zamiast deklarowania tekstu '.120EQ' - dla wypisania 120 znaków równości - lepiej użyć argumentu instrukcji DISPLAY z replikatorem:

(120)'.EQ'. Pierwsza postać zapisu powoduje zajęcie 120 miejsc pamięci - druga 2 (1 na replikator i 1 na wypisywany znak).

Dla uzyskania możliwie największej szybkości wykonania programu, szczególnie istotne znaczenie ma umiejętna segmentacja programu, tj. umieszczenie najczęściej używanych procedur w segmencie stałym i logiczne grupowanie procedur w segmentach wymiennych, tak aby stosunkowo czasochłonny proces wymiany segmentów w pamięci odbywał się możliwie rzadko. Wpływ na szybkość wykonania programu ma również technika czytania danych z taśmy papierowej. Najszyb-

sza i jednocześnie zajmująca najmniej miejsca w pamięci jest technika pozycyjna, ale wymaga specjalnego (czasem dość żmudnego) przygotowania danych. Programista powinien więc rozważyć co jest ważniejsze w danym przypadku: szybkość pracy programu, czy prostszy sposób przygotowania danych.

Podobnych przykładów można przytoczyć wiele. Ogólnie jednak należy stwierdzić, że dobra znajomość języka COBOL i wszystkich jego właściwości pozwala programiście pisać programy możliwie optymalne.

5.4.3. URUCHAMIANIE PROGRAMÓW

Po napisaniu programu wg przygotowanej sieci działań należy go uruchomić, tj. sprawdzić poprawność jego działania. Język COBOL daje wiele środków umożliwiających programiście szybkie i sprawne uruchamianie programów. Należą do nich przede wszystkim dyrektywy języka operacyjnego i instrukcje pomocnicze READY TRACE i RESET TRACE.

Dyrektywa PAM jest pomocna przy planowaniu pamięci, a szczególnie wtedy, gdy program nie mieści się w pamięci operacyjnej maszyny. Wiedząc bowiem ile miejsc pamięci "brakuje" i znając długość podprogramów realizujących działanie instrukcji (dodatek D), łatwiej jest optymalnie dokonać segmentacji programu.

Użycie dyrektywy TAB jest wygodne przy usuwaniu błędów formalnych i syntaktycznych zaistniałych w trakcie perforowania tekstu programu na taśmie papierowej lub kartach (szczególnie

w przypadkach gdy niesprawne urządzenie perforujące produkuje tabulogram różniący się od zawartości nośnika papierowego).

W wykrywaniu błędów logicznych w części proceduralnej bardzo pomocne są dyrektywy MON i SLA oraz dyrektywa TRA w połączeniu z instrukcją READY TRACE, umożliwiające śledzenie "drogi" programu.

Po uruchomieniu programu usunięcie z czołówki języka COBOL dyrektywy TRA likwiduje automatycznie wykonanie instrukcji READY TRACE zawartych w programie źródłowym. Przy uruchamianiu programu można również korzystać z innych instrukcji, np. wypisywać wyniki pośrednie instrukcją DISPLAY.

LISTA SŁÓW ZASTRZEŻONYCH JĘZYKA COBOL
=====Dla IDENTIFICATION DIVISION

DIVISION
IDENTIFICATION
PROGRAM..ID

Dla ENVIRONMENT DIVISION

APPLY	POSITIONAL
CONFIGURATION	RECORD
CONTINUATION	RECORDS
COPY	REEL
CURRENCY	RERUN
DELETION	RESERVE
DIVISION	SAME
ENVIRONMENT	SECTION
FILE	SELECT
FILE..CONTROL	SOURCE..COMPUTER
ITEM	SPECIAL..NAMES
I..O..CONTROL	STANDARD
INPUT	SWITCH
INPUT..OUTPUT	UNPOSITIONAL
MEMORY	ZAM..41
MTAPE	
MULTIPLE	
NO	
OBJECT..COMPUTER	
OFF	
ON	
OR	
OUTPUT	

D1a DATA DIVISION

BLANK	PIC
BLOCK	PICTURE
COPY	RECORD
DATA	RECORDS
DIVISION	REDEFINES
FD	RIGHT
FILE	SD
FILLER	SECUR
FIXED	SECT
FIXED2	SECTION
FLOAT	STANDARD
GENER	THRU
IDENT	VALUE
JUSTIFIED	VALUES
LABEL	WORKING..STORAGE
OCCURS	ZERO
OMITTED	

D1a PROCEDURE DIVISION

ACCEPT	COBOL
AFTER	COMP
ALL	COMPUTE
ALPHABETIC	COPY
ALTER	CREAT
AND	DECLARATIVES
ASCENDING	DEPENDING
BEFORE	DESCENDING
BEGINNING	DISPLAY
BUFF	DIVISION
BY	DRUM
CLOSE	ELSE

END	LOW..VALUE
ENDING	MOVE
ENTER	NEXT
EQUAL	NO
ERROR	NOT
EXAMINE	NOTE
EXIT	NUMERIC
EXPIR	OF
FILE	OFF
FIRST	ON
GENER	OPEN
GIVING	OR
GO	OUTPUT
GREATER	PERFORM
HARDWARE	PROCEDURE
HARDWARE ¹	PROGRAM
.	READ
.	READY
.	REEL
HARDWARE ⁿ	RELEASE
HIGH..VALUE	REPLACING
IDENT	RESET
IF	RETURN
IN	REWIND
INPUT	ROUNDED
LABEL	RUN
LEADING	SAS
LESS	SECT
LOCK	SECTION

SIZE
SORT
SPACE
STOP
TALLY
TALLYING
THEN
THRU
TIMES
TO
UNTIL
USE
USING
WRITE
ZERO

ALFABETYCZNY SPIS ZDAŃ, INSTRUKCJI ORAZ
ZWROTÓW JEZYKA COBOL DLA MASZINY ZAM 41

	str.		str.
ACCEPT	 204	MOVE	 208
ALTER	 217	MULTIPLE	 114
APPLY	 107	NOTE	 182
BLANK	 147	OCCURS	 149
BLOCK	 132	OPEN	 191
CLOSE	 199	PERFORM	 218
COMPUTE	 207	PICTURE	 136
COPY	 189	READ	 195
CURRENCY	 98	READY TRACE	 229
DATA	 128	REDEFINES	 150
DISPLAY	 206	RECORD	 133
END	 191	RELEASE	 227
ENTER	 181	RERUN	 115
EXAMINE	 212	RESET TRACE	 230
EXIT	 183	RETURN	 227
FIXED	 134	SELECT	 100
FIXED2	 134	SORT	 221
FLOAT	 134	STOP	 220
GO TO	 214	USE	 184
IF	 215	VALUE	 148
JUSTIFIED	 147	VALUE OF	 130
LABEL	 129	WRITE	 198

THE UNIVERSITY OF CHICAGO

LIBRARY

100	100	100	100
101	101	101	101
102	102	102	102
103	103	103	103
104	104	104	104
105	105	105	105
106	106	106	106
107	107	107	107
108	108	108	108
109	109	109	109
110	110	110	110
111	111	111	111
112	112	112	112
113	113	113	113
114	114	114	114
115	115	115	115
116	116	116	116
117	117	117	117
118	118	118	118
119	119	119	119
120	120	120	120
121	121	121	121
122	122	122	122
123	123	123	123
124	124	124	124
125	125	125	125
126	126	126	126
127	127	127	127
128	128	128	128
129	129	129	129
130	130	130	130
131	131	131	131
132	132	132	132
133	133	133	133
134	134	134	134
135	135	135	135
136	136	136	136
137	137	137	137
138	138	138	138
139	139	139	139
140	140	140	140
141	141	141	141
142	142	142	142
143	143	143	143
144	144	144	144
145	145	145	145
146	146	146	146
147	147	147	147
148	148	148	148
149	149	149	149
150	150	150	150

ZASADY DOŁĄCZANIA MODUŁÓW DO PROGRAMU

WYNIKOWEGO

Instrukcje języka COBOL są tłumaczone na ciągi rozkazów maszyny. Część instrukcji realizowana jest za pomocą podprogramów zwanych tutaj modułami.

Zasady dołączania modułów do segmentu programu wynikowego:

a) segment stały -

do segmentu stałego dołączane są zawsze moduły 88 (GO TO) i 46 (procedura monitorująca) oraz wszystkie te moduły, które są potrzebne w segmencie stałym programu wynikowego,

b) segment wymienny -

do segmentu wymiennego dołączane są te moduły, które są potrzebne w tym segmencie wymiennym, a nie zostały dołączone do segmentu stałego.

Każdy potrzebny moduł jest do segmentu dołączany tylko jeden raz.

Poniżej podane są dwie tabele. Pierwsza z nich podaje moduły dołączane do programu roboczego w zależności od rodzaju operacji występujących w programie źródłowym. Druga tabela podaje długości modułów liczone w słowach maszyny. Tabele te umożliwiają programiście przybliżone oszacowanie wielkości programu roboczego.

WYKAZ MODUŁÓW

WYKORZYSTYWANYCH W POSZCZEGÓLNYCH OPERACJACH
PROGRAMU ŹRÓDŁOWEGO

Lp.	Pierwsze słowo zdania lub instrukcji	Szczegółowe określenie przypadku.	Lista używanych modułów
1	2	3	4
1	ACCEPT	Argument typu FIXED	80,62,74, 30,39,43
2	ACCEPT	Argument typu FIXED2	80,62,79, 43,39,30
3	ACCEPT	Argument typu FLOAT	80,62,75, 30
4	ACCEPT	Argument opisany obrazem liczbowym	80,62,73, 30,35,31
5	ACCEPT	Argument opisany obrazem tekstowym	80,62,76, 30
6	ACCEPT	Argument opisany obrazem liczbowym wydawniczym	80,62,56, 40,53,60
7	ACCEPT	Argument opisany obrazem tekstowym wydawniczym	80,62,77, 40
8	ALTER		90
9	CLOSE	Nośnik - TM	33,70,63, 83,47,89, 66,68,32 zob.poz.55
10	CLOSE	Nośnik papierowy	70,89,49
11	COMPUTE	Operacja potęgowania(**)przy podstawie typu FIXED	86
12	COMPUTE	Operacja potęgowania(**)przy podstawie typu FIXED2	69,87:

1	2	3	4
13	COMPUTE	Operacja potęgowania(**) przy podstawie typu FLOAT	67
14	COMPUTE	Operacja mnożenia(*) 2 argumentów typu FIXED2	69
15	COMPUTE	Operacja dzielenia (/) dwóch argumentów typu FIXED2	59
16	COMPUTE	Operacja przesłania wartości wyrażenia arytmetycznego typu FIXED2 do pozycji opisanej obrazem liczbowym	zob. poz.28
17	COMPUTE	Operacja przesłania wartości wyrażenia arytmetycznego typu FIXED do pozycji opisanej obrazem liczbowym wydawniczym	zob. poz.37
18	COMPUTE	Operacja przesłania wartości wyrażenia arytmetycznego typu FIXED2 do pozycji opisanej obrazem liczbowym	zob. poz.29
19	COMPUTE	Operacja przesłania wartości wyrażenia arytmetycznego typu FIXED2 do pozycji opisanej obrazem liczbowym wydawniczym	zob. poz.38
20	COMPUTE	Operacja przesłania wartości wyrażenia arytmetycznego typu FLOAT do pozycji opisanej obrazem liczbowym	zob. poz.30
21	COMPUTE	Operacja przesłania wartości wyrażenia arytmetycznego typu FLOAT do pozycji opisanej obrazem liczbowym wydawniczym	zob. poz.39

1	2	3	4
22	DISPLAY		61
23	EXAMINE		58
24	EXIT	Operacja ta jest generowana przez translator na końcu paragrafu i sekcji	92
25	GO TO		88 (moduł ten jest dołączany zawsze do segmentu stałego)
26	IF	Porównywanie wielkości operandów znakovych	81
27	IF	Sprawdzenie klasy	82
28	MOVE	Pozycja źródłowa typu FIXED, pozycja wynikowa opisana obrazem liczbowym	71,31,30
29	MOVE	Pozycja źródłowa typu FIXED2, pozycja wynikowa opisana obrazem liczbowym	78,31,30
30	MOVE	Pozycja źródłowa typu FLOAT, pozycja wynikowa opisana obrazem liczbowym	72,31,30
31	MOVE	Pozycja źródłowa opisana obrazem liczbowym, pozycja wynikowa FIXED	74,30,39,43
32	MOVE	Pozycja źródłowa opisana obrazem liczbowym, pozycja wynikowa typu FIXED2	79,43,30,39
33	MOVE	Pozycja źródłowa opisana obrazem liczbowym, pozycja wynikowa typu FLOAT	75,30
34	MOVE	Pozycja źródłowa opisana obrazem liczbowym, pozycja wynikowa opisana obrazem liczbowym	73,30,31,35

1	2	3	4
35	MOVE	Pozycja źródłowa opisana obrazem znakowym, pozycja wynikowa opisana obrazem tekstowym	76,30
36	MOVE	Pozycja źródłowa opisana obrazem znakowym, pozycja wynikowa opisana obrazem tekstowym wydawniczym	77,40
37	MOVE	Pozycja źródłowa typu FIXED, pozycja wynikowa opisana obrazem liczbowym wydawniczym	30,40,54, 55,53,60
38	MOVE	Pozycja źródłowa typu FIXED2, pozycja wynikowa opisana obrazem liczbowym wydawniczym	30,40,55, 53,60
39	MOVE	Pozycja źródłowa typu FLOAT, pozycja wynikowa opisana obrazem liczbowym wydawniczym	72,30,31, 40,54,60, 56,53
40	MOVE	Pozycja źródłowa opisana obrazem liczbowym, pozycja wynikowa opisana obrazem liczbowym wydawniczym	56,53,40, 60
41	OPEN	Nośnik - TM	47,70,33, 68,89,63
42	OPEN	Nośnik papierowy	70,89,51
43	PERFORM		91
44	READ	Nośnik - TM	84,47,66, 32,89,64 oraz zob. poz.55
45	READ	Nośnik papierowy, czytanie pozycyjne	47,89,44 oraz zob. poz.55

1	2	3	4
46	READ	Nośnik papierowy, czytanie niepozycyjne	41,47,73,74,75, 76,79,89,30,31, 35,39,43,44 oraz zob.poz. 55
47	RELEASE		38,36,93,57 oraz moduły WRITE dla TM (zob.poz. 53)
48	RETURN		37,42,38,57 oraz moduły READ dla TM (zob. poz.44)
49	SORT	Opcje USING i GIVING	Moduły używane przez READ WRITE, OPEN,CLOSE dla TM, 42,38,93,57, 45,36 oraz moduły odpowiednich READ,OPEN,CLOSE, 52,37 oraz moduły odpowiednich WRITE,OPEN,CLOSE
50	SORT	Opcje USING i OUTPUT	Moduły używane przez READ,WRITE, OPEN,CLOSE dla TM, 42,38,93,57, 45,36 moduły odpowiednich READ, OPEN,CLOSE, 91
51	SORT	Opcje INPUT i GIVING	Moduły używane przez READ,WRITE, OPEN,CLOSE dla TM, 42,38,93,57, 52,37 oraz moduły odpowiednich WRITE,OPEN,CLOSE, 91
52	SORT	Opcje INPUT i OUTPUT	Moduły używane przez READ,WRITE, OPEN,CLOSE dla TM, 42,38,93,57, 91
53	WRITE	Nośnik - TM	83,68,47,66,32, 89,65 oraz zob. poz. 55
54	WRITE	Nośnik papierowy	70,89,49

1	2	3	4
55	RERUN	Zbiór modułów dołączanych dla odpowiedniej operacji WE-WY gdy do zbioru odnosi się zwrot RERUN	50,83,33,66, 47,89,68,70, 63,84,32,85

WYKAZ DŁUGOŚCI POSZCZEGÓLNYCH MODUŁÓW

Nr modułu	Długość w słowach maszyny	Nr modułu	Długość w słowach maszyny
1	2	3	4
30	36	62	8
31	119	63	494
32	22	64	127
33	35	65	112
34	325	66	314
35	53	67	33
36	107	68	29
37	62	69	25
38	94	70	41
39	18	71	32
40	14	72	81
41	465	73	78
42	30	74	20
43	64	75	120
44	300	76	43
45	35	77	53
46	189	78	32
47	23	79	19
48	41	80	252
49	39	81	60
50	285	82	62
51	197	83	93
52	52	84	141
53	382	85	206
54	38	86	21
55	45	87	25
56	62	88	42
57	53	89	27
58	45	90	25
59	50	91	35
60	17	92	16
61	45	93	91

MONITY CZASU TŁUMACZENIA
=====

W większości przypadków właściwy tekst błędu poprzedzany jest informacją postaci:

$\langle \text{nazwa-sekcji} \rangle \left\{ \begin{array}{l} \langle \text{nazwa-paragrafu} \rangle \\ \langle \text{nazwa-zbioru} \rangle \end{array} \right\} \langle \text{numer-zdania} \rangle$

W poszczególnych błędach informację tę będziemy dalej oznaczali "wsp" (współrzedne błędu).

$\langle \text{nazwa-sekcji} \rangle$ jest nazwą sekcji, w której występuje aktualnie sygnalizowany błąd.

$\langle \text{nazwa-paragrafu} \rangle$ jest nazwą paragrafu, w którym występuje błąd.

Gdy błąd występuje w FILE SECTION części DATA DIVISION, która nie ma struktury paragrafowej, wówczas zamiast nazwy paragrafu wypisywana jest $\langle \text{nazwa-zbioru} \rangle$.

Zamiast $\langle \text{nazwy-sekcji} \rangle$ wypisywane są trzy gwiazdki, jeśli błąd występuje w części programu, która nie jest zawarta w sekcji.

Zamiast $\langle \text{nazwy-paragrafu} \rangle$ względnie $\langle \text{nazwy-zbioru} \rangle$ wypisywane są trzy gwiazdki, jeśli błąd występuje w części programu nie zawartej ani w paragrafie, ani w opisie zbioru.

W czasie tłumaczenia programu poszczególne jego zdania są numerowane. Numeracja zdań zaczyna się od jedności w każdym paragrafie. Poniżej przez $\langle \text{numer-zdania} \rangle$ będziemy rozumieli numer kolejny tego zdania w paragrafie, w którym ono występuje. Jeżeli w części PROCEDURE DIVISION nie występują paragrafy, to numeracja

zdań zaczyna się od każdej nowej sekcji, a jeżeli nie występują również sekcje, to w obrębie całej części PROCEDURE obowiązuje jednolita numeracja zdań.

Pierwszy błąd występujący w PROCEDURE DIVISION, w pewnej wstawce SAS, poprzedzany jest informacją <współrządne-błądu>. Pozostałe błędy w tej samej wstawce nie są poprzedzane taką informacją. <numer-wiersza> w monitach o błędach we wstawce oznacza kolejny numer nie pustego wiersza danej wstawki.

W czasie tłumaczenia programu sygnalizowane są najwyżej 32 błędy, po czym następuje przerwanie pracy tłumacza.

Teksty błędów czasu tłumaczenia programu wypisywane są na zerowym wyjściu tłumacza. W niniejszym opisie teksty te uporządkowane są alfabetycznie.

BŁOK NIE RERUNOWY

Znaczenie: blok, którego numer podany jest w czołówce tłumacza RESTART, nie jest blokiem RERUN.

Reakcja tłumacza: przerwanie działania tłumacza RESTART.

BŁĘDNA CZOŁÓWKA PROGRAMU COBOL

Znaczenie: w czołówce programu COBOL występuje niepoprawny znak lub słowo.

Reakcja tłumacza: tłumaczenie programu zostaje przerwane.

BŁĘDNA CZOŁÓWKA RESTART

Znaczenie: błędna czołówka translatora RESTART.

Reakcja translatora: przerwanie działania translatora RESTART.

wsp BLEDNA NAZWA DANEJ x

x - <nazwa-danej>

Znaczenie: w PROCEDURE DIVISION, w zdaniu <numer-zdania> występuje <nazwa-danej>, nie zadeklarowana albo niejednoznacznie zadeklarowana w DATA DIVISION.

Reakcja translatora: całe zdanie <numer-zdania> zawierające błędną <nazwę-danej> jest opuszczane. Tłumaczenie programu jest kontynuowane.

wsp BLEDNA STRUKTURA x

x - <błędny-kontekst>

Znaczenie: struktura fragmentu programu jest niezgodna ze składnią języka COBOL (np. w PROCEDURE DIVISION niektóre zdania występują w sekcjach, a inne nie).

Reakcja translatora: tłumaczenie programu jest przerywane.

wsp BLEDNE VALUE OF

Znaczenie: pozycji metryki początku zbioru nadawana jest niewłaściwa wartość.

Reakcja translatora: pozycji metrykowej nadawana jest wartość standardowa. Zwrot jest pomijany, tłumaczenie programu jest kontynuowane.

BLEDNY OGRANICZNIK

Znaczenie: pojawienie się fizycznego końca programu przed END PROGRAM. Pośrednio może to być spowodowane brakiem ogranicznika tekstu lub komentarza.

Reakcja translatora: tłumaczenie programu zostaje przerwane.

wsp BLEDNY POZATEK INFORMACJI x y

x -<numer-wiersza>
y -<zawartość-wiersza>

Znaczenie: na początku wiersza <numer-wiersza> we wstawce SAS wystąpił znak (lub * .

Reakcja translatora: błędnie napisany wiersz jest opuszczany. Tłumaczenie programu jest kontynuowane.

wsp BLEDNY TEKST x y

x -<numer-wiersza>
y -<zawartość-wiersza>

Znaczenie: we wstawce SAS, w wierszu <numer-wiersza> występuje niepoprawny symbol specjalny w tekście.

Reakcja translatora: błędny tekst jest pomijany; tłumaczenie programu jest kontynuowane.

wsp<numer-zdania>BLEDNY ZNAK

Znaczenie: w zwrocie APPLY UNPOSITIONAL, w którym określone są niestandardowe wartości ograniczników, znaku przeniesienia, znaku wymazania lub znaku, z którego składa się oznacznik punktu ponawiania, jedna z tych niestandardowych wartości jest tekstem zawierającym więcej niż jeden znak.

Reakcja translatora: wielkości, której niestandardowa wartość jest błędna, przypisywana jest wartość standardowa; tłumaczenie programu jest kontynuowane.

wsp BLEDNY ZWROT GRUPOWEJ

Znaczenie: w opisie pozycji grupowej występuje zwrot, który może opisywać wyłącznie pozycje elementarne (np. zwrot PICTURE).

Reakcja translatora: błędny zwrot jest pomijany.

BRAK DEKLARACJI PARAGRAFU x W SEKCJI y

x -<nazwa-paragrafu>

y -<nazwa-sekcji>

Znaczenie: w PROCEDURE DIVISION występuje odwołanie do paragrafu <nazwa-paragrafu> zawartego w sekcji <nazwa-sekcji>, którego nie ma w tej sekcji.

Reakcja translatora: instrukcja zawierająca nazwę nie istniejącego paragrafu zostaje zaadresowana w sposób losowy. Tłumaczenie programu jest kontynuowane.

BRAK DEKLARACJI SEKCJI x

x -<nazwa-sekcji>

Znaczenie: w PROCEDURE DIVISION występuje odwołanie do sekcji, która nie została zadeklarowana w tej części.

Reakcja translatora: instrukcja zawierająca nazwę nieistniejącej sekcji zostaje zaadresowana w sposób losowy. Tłumaczenie programu jest kontynuowane.

wsp BRAK LABEL '

Znaczenie: w opisie zbioru w FILE SECTION części DATA DIVISION nie występuje zwrot LABEL.

Reakcja translatora: przyjmuje, że wystąpił zwrot LABEL RECORD OMITTED i kontynuuje tłumaczenie programu.

wsp BRAK LUB POWTORZENIE ZBIORU x

x -<nazwa-zbioru>

Znaczenie: a) w zwrocie SAME AREA lub SAME RECORD AREA występuje nazwa zbioru, który nie został opisany w DATA DIVISION,

b) wielokrotne użycie nazwy tego samego zbioru w kilku zwrotach SAME RECORD AREA.

Reakcja translatora: zwrot zawierający niewłaściwą nazwę zbioru jest pomijany. Tłumaczenie programu jest kontynuowane.

wsp BRAK PARAMETRU REZE x y

x -<numer-wiersza>

y -<zawartość-wiersza>

Znaczenie: w dyrektywie REZE we wstawce SAS nie występuje <całkowita>.

Reakcja translatora: niepoprawny wiersz jest pomijany. Tłumaczenie programu jest kontynuowane.

wsp BRAK SELECT

Znaczenie: do zbioru opisanego w DATA DIVISION nie odnosi się w ENVIRONMENT DIVISION (INPUT..OUTPUT SECTION) deklaracja SELECT.

Reakcja translatora: tłumaczenie jest kontynuowane, przy czym przyjmuje się, że do zbioru odnosił się zwrot SELECT <nazwa-zbioru>ASSIGN TO MTAPE A.

wsp BRAK TYPU

Znaczenie: w deklaracji pozycji elementarnej nie jest podane czy jest ona typu FIXED, FIXED2, FLOAT czy PICTURE.

Reakcja translatora: pozycji nadawany jest typ FIXED; tłumaczenie jest kontynuowane.

wsp BRAK ZMIENNEJ

Znaczenie: w DATA DIVISION nie jest opisana nazwa danej występująca w zwrocie REDEFINES.

Reakcja translatora: błędny zwrot REDEFINES jest pomijany. Tłumaczenie programu jest kontynuowane.

wsp ETYKIETA PARAMETREM FORMALNYM x y

x -<numer-wiersza>
y -<zawartość-wiersza>

Znaczenie: we wstawce SAS w wierszu <numer-wiersza> występuje deklaracja takiej etykiety, która występuje również na liście parametrów formalnych tej wstawki (tj. w zwrocie REPLACING BY).

Reakcja translatora: wiersz <numer-wiersza> wstawki jest opuszczany. Tłumaczenie programu jest kontynuowane.

wsp NADMIAR ADRESU RZECZYWISTEGO x y

x -<numer-wiersza>
y -<zawartość-wiersza>

Znaczenie: w wierszu <numer-wiersza> wstawki SAS

- a) adres rzeczywisty jest większy od 2^{15} ,
- b) składowa adresu jest większa od 2^{23} .

Reakcja translatora: niepoprawny wiersz jest pomijany. Tłumaczenie programu jest kontynuowane.

wsp NADMIAR LICZBY x y

x -<numer-wiersza>
y -<zawartość-wiersza>

Znaczenie: w wierszu <numer-wiersza> wstawki SAS występuje za duża liczba.

Reakcja translatora: błędna liczba jest opuszczana. Tłumaczenie programu jest kontynuowane.

wsp NIE DOLACZONY ZBIOR BIBLIOTECZNY x

x -<nazwa-zbioru-bibliotecznego>

Znaczenie: w zdaniu COPY występuje <nazwa-zbioru-bibliotecznego>, która nie wystąpiła na liście dyrektywy BIB w czołówce programu.

Reakcja translatora: tłumaczenie programu zostaje zatrzymane.

wsp NIEDOZWOLONY ZNAK x y

x -<numer-wiersza>

y -<zawartość-wiersza>

Znaczenie: na początku wiersza <numer-wiersza> wstawki SAS lub po deklaracji etykiety w tym wierszu wystąpił znak nie należący do zbioru (litera cyfra) = + - ' / spacja nowa-linia.

Reakcja translatora: wiersz zawierający błędny znak jest pomijany. Tłumaczenie programu jest kontynuowane.

wsp /NIEJEDNOZNACZNA NAZWA |

Znaczenie: nazwa danej występująca w części DATA DIVISION w zwrocie REDEFINES jest w tej części niejednoznacznie opisana.

Reakcja translatora: błędny zwrot REDEFINES jest omijany. Tłumaczenie programu przebiega dalej. .

wsp NIEPOPRAWNA CZESC ADRESOWA x y

x -<numer-wiersza>

y -<zawartość-wiersza>

Znaczenie: w wierszu <numer-wiersza> wstawki SAS występuje niepoprawna część adresowa.

Reakcja translatora: niepoprawny wiersz jest pomijany, a tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNA LICZBA x y

x -<numer-wiersza>

y -<zawartość-wiersza>

Znaczenie: w wierszu <numer-wiersza> wstawki SAS występuje niepoprawna liczba, tj.

- a) na początku wiersza albo po deklaracji etykiety za znakiem + lub - występuje inny znak niż cyfra,
- b) suma ilości cyfr liczby przed i po kropce jest równa zero,
- c) po literze E w liczbie zmiennoprzecinkowej nie występuje wykładnik,

- d) liczba nie jest poprzedzona znakiem + lub - .

Reakcja translatora: błędna liczba jest pomijana. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNA OPERACJA POTEGOWANIA

Znaczenie: a) podstawa albo wykładnik potęgi nie jest typu obliczeniowego,

b) wykładnik potęgi nie jest typu FIXED.

Reakcja translatora: instrukcja zawierająca niepoprawną potęgę oraz wszystkie instrukcje występujące za nią w zdaniu <numer-zdania> są pomijane, a tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNE PRZESŁANIE

Znaczenie: instrukcja MOVE napisana niezgodnie z regułami dotyczącymi przesłań.

Reakcja translatora: fragment zdania <numer-zdania>, począwszy od błędnej instrukcji MOVE do końca zdania, jest opuszczany. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT ACCEPT

Znaczenie: nazwa danej występująca w instrukcji ACCEPT nie jest nazwą pozycji elementarnej.

Reakcja translatora: fragment programu od błędnej instrukcji ACCEPT do końca zdania <numer-zdania>, w którym ta instrukcja występuje, jest opuszczany. Tłumaczenie programu jest kontynuowane.

NIEPOPRAWNY ARGUMENT ALTER LUB PERFORM W SEGMENTCIE x

x - <numer-segmentu>

Znaczenie: argument instrukcji ALTER albo PERFORM, zawartej w PROCEDURE DIVISION w segmencie wymiennym <numer-segmentu>, dotyczy innego segmentu wymiennego.

Reakcja translatora: błędna instrukcja ALTER albo PERFORM zostaje zaadresowana w sposób losowy.

wsp NIEPOPRAWNY ARGUMENT CLOSE

Znaczenie: argument instrukcji CLOSE nie jest nazwą zbioru.

Reakcja translatora: fragment programu począwszy od błędnej instrukcji CLOSE do końca zdania <numer-zdania> jest opuszczany. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT DISPLAY

Znaczenie: a) nazwa danej w instrukcji DISPLAY nie jest nazwą pozycji elementarnej,

b) w instrukcji występuje nazwa pozycji elementarnej typu FIXED, FIXED2 lub FLOAT,

- c) wartość <wyrażenia-arytmetycznego> występującego w instrukcji nie jest typu FIXED.

Reakcja tłumacza: błędna instrukcja DISPLAY oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT ENTER

Znaczenie: argument zwrotu REPLACING w zdaniu ENTER nie jest nazwą elementarnej albo grupowej jednostki danych.

Reakcja tłumacza: zdanie ENTER jest opuszczane. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT EXAMINE

Znaczenie: nazwa danej w instrukcji EXAMINE nie jest nazwą pozycji elementarnej albo jest nazwą pozycji typu FIXED, FIXED2 lub FLOAT.

Reakcja tłumacza: błędna instrukcja EXAMINE oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT GO TO DEPENDING

Znaczenie: nazwa danej w instrukcji GO TO z frazą DEPENDING nie jest nazwą pozycji typu FIXED.

Reakcja translatora: błędna instrukcja GO TO oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT PERFORM TIMES

Znaczenie: argument frazy TIMES w instrukcji PERFORM nie jest typu FIXED.

Reakcja translatora: błędna instrukcja PERFORM oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane, a tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT OPEN

- Znaczenie:
- a) argument instrukcji OPEN nie jest nazwą zbioru,
 - b) zbiorowi występującemu w instrukcji OPEN INPUT przydzielono urządzenie wyjściowe,
 - c) zbiorowi występującemu w instrukcji OPEN OUTPUT przydzielono urządzenie wejściowe.

Reakcja translatora: błędna instrukcja OPEN oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT READ

- Znaczenie:
- a) argument instrukcji READ nie jest nazwą zbioru,

- b) argument instrukcji READ jest nazwą zbioru, któremu za pomocą deklaracji SELECT przydzielono urządzenie wyjściowe.

Reakcja translatora: błędna instrukcja READ oraz wszystkie występujące po niej w zdaniu <numer-zdania> są pomijane, tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT RELEASE

Znaczenie: argument instrukcji RELEASE nie jest nazwą dokumentu sortowania.

Reakcja translatora: błędna instrukcja RELEASE oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane, a tłumaczenie programu przebiega dalej.

wsp NIEPOPRAWNY ARGUMENT RETURN

Znaczenie: argument instrukcji RETURN nie jest nazwą zbioru sortowania.

Reakcja translatora: błędna instrukcja RETURN oraz wszystkie instrukcje występujące za nią w zdaniu <numer-zdania> są pomijane, a tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT SORT

Znaczenie: a) identyfikator występujący po słowie SORT nie jest nazwą zbioru sortowania,

- b) nazwa klucza nie jest zmienną elementarną,

- c) identyfikator występujący w zwrocie USING nie jest nazwą zbioru wejściowego, .
- d) identyfikator występujący w zwrocie GIVING nie jest nazwą zbioru wyjściowego.

Reakcja translatora: błędna instrukcja SORT oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane, a tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT SPRAWDZENIA KLASY

- .Znaczenie:
- a) argument sprawdzenia klasy nie jest zmienną ani zmienną indeksowaną,
 - b) argument ALPHABETIC nie jest znakowy ani znakowy wydawniczy,
 - c) argument NUMERIC jest inny niż liczbowy, liczbowy wydawniczy, znakowy lub znakowy wydawniczy.

Reakcja translatora: instrukcja, w której występuje błędne sprawdzenie klasy oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY ARGUMENT WRITE

- Znaczenie: a) argument instrukcji WRITE nie jest nazwą dokumentu,

- b) argument instrukcji WRITE jest nazwą dokumentu należącego do zbioru, któremu za pomocą deklaracji SELECT przypisano urządzenie wejściowe.

Reakcja tłumacza: błędna instrukcja WRITE oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane, a tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY KONIEC WIERSZA x y

x - <numer-wiersza>

y - <zawartość-wiersza>

Znaczenie: poprawny syntaktycznie wiersz <numer-wiersza> we wstawce SAS zakończony jest niewłaściwym znakiem.

Reakcja tłumacza: wiersz jest opuszczany. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY INDEKS

Znaczenie: wynik obliczenia wyrażenia arytmetycznego będącego indeksem nie jest typu FIXED.

Reakcja tłumacza: instrukcja, w której występuje błędnie napisany indeks oraz wszystkie instrukcje występujące za nią w zdaniu <numer-zdania> są pomijane, a tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY OPERAND ARYTMETYCZNY

Znaczenie: operand operacji arytmetycznej nie jest obliczeniowy (tj. albo nie jest

elementarny albo jest elementarny ale nie typu FIXED, FIXED2 lub FLOAT).

Reakcja translatora: instrukcja zawierająca błędnie napisane wyrażenie arytmetyczne oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane, a tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY OPERAND LOGICZNY

Znaczenie: operand występujący w warunku nie jest typu logicznego.

Reakcja translatora: fragment programu od błędnej instrukcji IF bądź PERFORM do końca zdania <numer-zdania>, w którym ta instrukcja występuje, jest opuszczany. Tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY OPERAND POROWNANIA

Znaczenie: a) operand porównania wielkości nie jest elementarny,
b) operand porównania nie jest ani obliczeniowy, ani znakowy.

Reakcja translatora: instrukcja zawierająca niepoprawne porównanie wielkości oraz wszystkie instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane i tłumaczenie programu jest kontynuowane.

wsp NIEPOPRAWNY SZABLON x y

x -<numer-wiersza>

y -<zawartość-wiersza>

Znaczenie: w wierszu <numer-wiersza> we wstawce SAS występuje niepoprawny szablon, tj. a) po znaku = występuje inny znak niż cyfra,

b) po znaku = występuje cyfra 8 lub 9,

c) szablon jest większy niż
=7777777777777777.

Reakcja translatora: niepoprawny wiersz jest pomijany, a tłumaczenie programu jest kontynuowane.

wsp. NIEWŁĄSCIWE REDEFINES

Znaczenie: zwrot REDEFINES opisuje taką zmienną, do której ten zwrot nie może się odnosić.

Reakcja translatora: błędnie użyty zwrot REDEFINES jest opuszczany. Tłumaczenie programu jest kontynuowane.

wsp. NIEWŁĄSCIWE URZADZENIE

Znaczenie: a) urządzenie przydzielone zbiorowi w deklaracji SELECT nie jest zgodne z techniką określoną dla tego zbioru w zwrocie APPLY,

b) zbiorowi sortowania przydzielono inne urządzenie niż taśmę magnetyczną. <współrzedne-błądu> wskazują zdanie zawierające FD albo SD danego zbioru.

Reakcja translatora: tłumaczenie programu jest kontynuowane. Zbiorowi danych nie jest przydzielane konkretne urządzenie, ale jeśli w zwrocie APPLY odnoszącym się do tego zbioru występuje

- a) fraza STANDARD, to dalej zbiór traktuje się jako taśmowy, *
- b) fraza POSITIONAL bądź UNPOSITIONAL, to dalej zbiór traktuje się jako wejściowy, związany z symbolicznym wejściem-wyjściem.

wsp NIEWŁASCIWE VALUE

Znaczenie: wartość początkowa, przypisana zmiennej zwrotem VALUE, jest niezgodna z typem zmiennej, jej długością, skalą bądź znakowaniem.

Reakcja translatora: błędny zwrot VALUE jest pomijany. Tłumaczenie programu jest kontynuowane.

wsp NIEWŁASCIWY RERUN

Znaczenie: w zwrocie RERUN opisującym zbiór związany z taśmą magnetyczną występuje fraza EVERY RERUN MARK <znak>.

Reakcja translatora: błędny zwrot RERUN jest pomijany i tłumaczenie programu jest kontynuowane.

wsp NIEWŁASCIWY TYP

Znaczenie: typ zmiennej niedopuszczalny dla operacji wejścia/wyjścia, w których

zmienna nawet pośrednio występuje;
<współrzędne-błędu> wskazują opis tej
/zmiennej w DATA DIVISION.

Reakcja translatora: zmienna otrzymuje własności wynikające z jej opisu. Tłumaczenie programu jest kontynuowane.

wsp NIEZADEKLAROWANA ETYKIETA x y

x -<numer-wiersza>

y -<zawartość-wiersza>

Znaczenie: w wierszu <numer-wiersza> we wstawce SAS, w adresie symbolicznym występuje etykieta, która nie została zadeklarowana we wstawce ani nie wystąpiła na liście parametrów formalnych tej wstawki.

Reakcja translatora: wiersz jest pomijany. Tłumaczenie programu jest kontynuowane.

wsp NIEZGODNOSC TYPOW OPERANDOW POROWNANIA

Znaczenie: w operacji porównania wielkości operandy są niezgodne co do typu.

Reakcja translatora: instrukcja zawierająca niepoprawne porównanie oraz instrukcje występujące po niej w zdaniu <numer-zdania> są pomijane. Tłumaczenie programu jest kontynuowane.

wsp NIEZNANA CZESC OPERACYJNA x y

x -<numer-wiersza>

y -<zawartość wiersza>

Znaczenie: w wierszu <numer-wiersza> wstawki SAS, na początku wiersza lub po deklaracji etykiety występuje ciąg literowo-cyfrowy, który nie jest ani częścią operacyjną rozkazu maszyny ZAM.41, ani dyrektywą REZE.

Reakcja translatora: niepoprawny wiersz jest pomijany, a tłumaczenie programu jest kontynuowane.

wsp PODWOJNA DEKLARACJA PARAGRAFU x

x -<nazwa-paragrafu>

Znaczenie: w PROCEDURE DIVISION w jednej sekcji wielokrotnie występuje deklaracja tej samej nazwy paragrafu.

Reakcja translatora: tłumaczenie programu jest kontynuowane.

wsp PODWOJNA DEKLARACJA SEKCJI x

x -<nazwa-sekcji>

Znaczenie: w PROCEDURE DIVISION wielokrotnie występuje deklaracja tej samej nazwy sekcji.

Reakcja translatora: tłumaczenie programu jest kontynuowane.

PRZEKLAMANIA TASMY

Znaczenie: przed wykonaniem programu od pewnego punktu ponawiania blok RERUN nie daje się odczytać z taśmy magnetycznej.

Reakcja translatora: przerwanie działania translatora RESTART.

PRZEPŁNIENIE BĘBNA - x

x - <etap=translacji> (zob. niżej)

Znaczenie: brak miejsca na bębnie dla pośrednich wyników translacji lub binarnego programu wynikowego. Moment przepełnienia określany jest przez etap translacji, który może przyjmować jedną z poniżej opisanych postaci:

- POSR - w czasie tworzenia wyników pośrednich translacji,
- STAŁE - w czasie zestawienia stałych programu wynikowego,
- PROC - w czasie zestawiania części proceduralnej programu wynikowego.

Reakcja translatora: tłumaczenie programu zostaje zatrzymane.

SEGMENT x NIE MIESCI SIE W PAO, BRAK y

x - <numer-segmentu>

y - <ilość-słów>

Znaczenie: segment <numer-segmentu> nie mieści się w pamięci operacyjnej. <ilość-słów> określa o ile słów segment przekracza wielkość PAO.

Reakcja translatora: tłumaczenie programu jest kontynuowane.

wsp VALUE DLA OBLICZENIOWEJ

Znaczenie: w opisie niepozycyjnego zbioru wejściowego występuje znacznik dokumentu typu FIXED, FIXED2 lub FLOAT.

Reakcja translatora: błędny zwrot VALUE jest pomijany, a tłumaczenie programu jest kontynuowane.

W BIBLIOTECE BRAK ZBIORU x

x -<nazwa-zbioru-bibliotecznego>

Znaczenie: w bibliotece systemu nie istnieje zbiór, którego nazwa występuje jako argument dyrektywy BIB.

Reakcja translatora: tłumaczenie programu zostaje przerwane.

wsp WIELOKROTNA DEKLARACJA ETYKIETY x y

x -<nazwa-wiersza>

y -<zawartość-wiersza>

Znaczenie: ta sama etykieta została w tej samej wstawce SAS, w wierszu <numer-wiersza>, zadeklarowana nie po raz pierwszy.

Reakcja translatora: błędna (nie pierwsza) deklaracja etykiety jest opuszczana wraz z zawierającym ją wierszem. Tłumaczenie programu jest kontynuowane.

ZA DLUGA WSTAWKA SAS

Znaczenie: brak miejsca w PAO na przetłumaczenie wstawki.

Reakcja translatora: tłumaczenie programu zostaje zatrzymane.

wsp ZA DŁUGI IDENT

Znaczenie: tekst, który ma być wartością pozycji metrykowej IDENT (zwrot VALUE OF) zawiera więcej niż 18 znaków.

Reakcja translatora: błędny zwrot VALUE OF IDENT posyłana jest wartość standardowa, tłumaczenie programu jest kontynuowane.

ZA DŁUGIE ZDANIE LUB ZA DUŻO PARAGRAFOW W SEKCJI

Znaczenie: brak miejsca w PAO na przetłumaczenie zbyt długiego zdania albo słownik nazw proceduralnych sekcji nie mieści się w PAO.

Reakcja translatora: tłumaczenie programu zostaje zatrzymane.

ZA DŁUGIE ZDANIE

Znaczenie: brak miejsca w PAO na przetłumaczenie zdania.

Reakcja translatora: tłumaczenie programu zostaje zatrzymane.

ZA DUŻA ILOSC SEGMENTOW - x

x - <etap translacji> (zob.opis błędu PRZEPENNIENIE BEBNA)

Znaczenie: brak miejsca w PAO na kompilację kolejnego segmentu.

Reakcja translatora: tłumaczenie programu zostaje zatrzymane.

ZA DUŻO ETYKIET WE WSTAWCE SAS

Znaczenie: lista etykiet wstawki nie mieści się w pamięci operacyjnej danego zestawu maszyny.

Reakcja translatora: tłumaczenie programu zostaje zatrzymane.

wsp ZA DUŻO KLUCZY

Znaczenie: w instrukcji SORT liczba kluczy przekracza 10.

Reakcja translatora: błędna instrukcja SORT jest pomijana. Tłumaczenie programu jest kontynuowane.

ZA DUŻO ZBIORÓW

Znaczenie: tworzona w czasie translacji lista zbiorów nie mieści się w pamięci operacyjnej danego zestawu maszyny.

Reakcja translatora: tłumaczenie programu zostaje zatrzymane.

wsp ZA DUŻY BLOK

Znaczenie: dla zbipru na taśmie magnetycznej zadeklarowano blok przekraczający 2047 słów.

Reakcja translatora: tłumaczenie programu jest kontynuowane.

ZA MAŁA PAO

Znaczenie: obszar wolnej pamięci operacyjnej dla programu, który ma być wykonywany od pewnego punktu ponawiania, jest mniejszy od długości bloku RERUN.

Reakcja translatora: przerwanie działania translatora RESTART.

wsp ZA MAŁO TASM

Znaczenie: zbiorowi sortowania przydzielono w odpowiedniej deklaracji SELECT za mało jednostek taśmy magnetycznej. <współrzędne-błędu> wskazują zdanie zawierające SD danego zbioru sortowania.

Reakcja translatora: tłumaczenie programu jest kontynuowane.

wsp ZBIOR BIBLIOTECZNY x BŁĘDNA NAZWA DANEJ |y

x - <nazwa-zbioru-bibliotecznego>

y - <nazwa-danej>

Znaczenie: w zbiorze bibliotecznym <nazwa-zbioru-bibliotecznego>, włączonym do programu za pomocą zdania COPY, występuje <nazwa-danej> nie zadeklarowana albo nie jednoznacznie zadeklarowana w DATA DIVISION.

Reakcja translatora: całe zdanie zawiera-
jące błędną nazwę danej jest opuszczane.
Tłumaczenie programu jest kontynuowane.

wsp ZBIOR BIBLIOTECZNY x
BŁĘDNA STRUKTURA y

x -<nazwa-zbioru-bibliotecznego>
y -<błędny-kontekst>

Znaczenie: w zbiorze bibliotecznym <nazwa-
zbioru-bibliotecznego> występuje fragment
niezgodny co do struktury ze składnią ję-
zyka COBOL (np. w DATA DIVISION numer po-
ziomu 77 występuje w FILE SECTION).

Reakcja translatora: tłumaczenie programu
jest przerywane.

wsp ZBIOR BIBLIOTECZNY x
ZŁA SKŁADNIA y

x -<nazwa-zbioru-bibliotecznego>
y -<błędny-kontekst>

Znaczenie: w zbiorze bibliotecznym <nazwa-
zbioru-bibliotecznego>, włączonym do pro-
gramu za pomocą zdania COPY, występuje
zdanie <numer-zdania> napisane niezgodnie
ze składnią języka COBOL.

Reakcja translatora: błędnie zbudowane
zdanie jest opuszczane. Tłumaczenie pro-
gramu jest kontynuowane.

wsp ZBIOR BIBLIOTECZNY x

ZLA SKLADNIA IF y

x -<nazwa-zbioru-bibliotecznego>

y -<błędny-kontekst>

Znaczenie: w zbiorze bibliotecznym <nazwa-zbioru-bibliotecznego>, włączonym do programu za pomocą zdania COPY, w zdaniu <numer-zdania> występuje zagnieżdżona instrukcja IF, w której liczba fraz THEN jest inna niż liczba fraz ELSE albo frazy te występują w błędnej kolejności.

Reakcja translatora: błędne zdanie jest opuszczane. Tłumaczenie programu jest kontynuowane.

wsp ZLA SKLADNIA x

x -<błędny-kontekst>

Znaczenie: zdanie <numer-zdania> jest napisane niezgodnie ze składnią języka COBOL (np. w wyrażeniu arytmetycznym liczba nawiasów otwierających jest inna niż liczba nawiasów zamykających).

Reakcja translatora: błędnie zbudowane zdanie jest opuszczane. Tłumaczenie programu jest kontynuowane.

wsp ZLA SKŁADNIA IF x

x -<błędny-kontekst>

Znaczenie: w zagnieżdżonej instrukcji IF liczba fraz THEN jest inna niż liczba fraz ELSE albo frazy te występują w błędnej kolejności.

Reakcja translatora: błędne zdanie jest opuszczane. Tłumaczenie programu jest kontynuowane.

ZLY OBRAZ: NR x y

x -<numer-kolejny>

y -<błędny-obraz>

Znaczenie: w opisie pozycji elementarnych w DATA DIVISION obraz o numerze <numer-kolejny> jest błędny. Pierwszy od początku obraz w DATA DIVISION ma <numer-kolejny> równy jedności, następny za nim - <numer-kolejny> dwa itd.

Reakcja translatora: błędny obraz zastępowany jest obrazem liczbowym '9' i tłumaczenie programu jest kontynuowane.

wsp ZLY ZWROT PICTURE: x

x -<błędny-obraz>

Znaczenie: argument zwrotu PICTURE nie jest poprawnym obrazem (np. brak apostrofów).

Reakcja translatora: błędny obraz zostaje zastąpiony obrazem liczbowym '9'. Tłumaczenie programu jest kontynuowane.

MONITY CZASU WYKONANIA
=====

Błędy i monity czasu wykonania programu wypisywane są na monitorze maszyny.

W poniższym spisie przyjęto następujące oznaczenia:

- przez <identyfikator-zbioru> rozumie się w poniższych tekstach wartość pozycji metrykowej IDENT, określoną dla danego zbioru w programie,
- przez <symbol-urządzenia> rozumie się
 - dla taśm magnetycznych jedną z liter: A, B, C, ... F przypisanych zbiorowi w deklaracji SELECT i określających numer kolejny stacji taśmy magnetycznej w konkretnym zestawie maszyny użytkowej,
 - dla zbiorów związanych z symbolicznym wejściem lub wyjściem - liczbę całkowitą będącą numerem symbolicznego wejścia lub symbolicznego wyjścia, przypisanego zbiorowi w deklaracji SELECT.

Alfabetyczny spis błędów i monitów czasu
wykonywania programu

AWARIA STACJI ZBIORU x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

Znaczenie: awaria stacji magnetycznej
<symbol-urządzenia> podczas wykonywania
instrukcji READ lub WRITE.

Reakcja programu: przerwanie wykonywania
programu.

BŁĄD I KONIEC TAŚMY ZBIORU x y z

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

z -<numer-bloku>

Znaczenie: błąd przy pisaniu bloku <numer-
bloku> na taśmę magnetyczną podczas wyko-
nywania instrukcji WRITE. Zasygnalizowany
zbliżający się koniec szpuli uniemożliwia
poprawienie błędnego zapisu.

Reakcja programu: przerwanie wykonywania
programu.

BŁĘDNE DANE ACCEPT x

x -<symbol-urządzenia>

Znaczenie: próba wykonania instrukcji ACCEPT,
gdy jednostka danych jest tekstowa, a pozy-
cja jest obliczeniowa albo liczbowa, albo
liczbowa wydawnicza.

Reakcja programu: następuje zawieszenie maszyny.

BRAK MIEJSCA DLA ZBIORU x y

x - <identyfikator-zbioru>

y - <symbol-urządzenia>

Znaczenie: sygnalizowany w trakcie wykonania instrukcji OPEN OUTPUT gdy

- a) szpula taśmy jest wielozbiorowa, a zbliżający się koniec szpuli uniemożliwia otwarcie na niej nowego zbioru do pisanie,
- b) zbiór jest jednoszpulowy, a na szpuli zapisany jest inny zbiór, który nie jest zdezaktualizowany.

Reakcja programu: następuje zawieszenie maszyny. Po jej odwieszeniu przez operatora instrukcja OPEN OUTPUT wykonywana jest ponownie.

BRAK ZBIORU x y

x - <identyfikator-zbioru>

y - <symbol-urządzenia>

Znaczenie: sygnalizowany w trakcie wykonywania instrukcji OPEN INPUT:

- a) dla zbiorów związanych z taśmą magnetyczną - gdy na szpuli taśmy nie można znaleźć metryki początku zbioru <identyfikator-zbioru>,
- b) dla zbiorów związanych z symbolicznym wejściem

- gdy metryka początku zbioru jest niepoprawnie zbudowana, np. nie zawiera identyfikatora, względnie daty utworzenia zbioru,
- gdy metryka jest zbudowana poprawnie ale zawiera inny identyfikator zbioru lub numer sekcji.

Reakcja programu: następuje zawieszenie maszyny. Po odwieszeniu maszyny przez operatora instrukcja OPEN INPUT wykonywana jest ponownie.

DZIELNIK ZERO x

x -<numer-segmentu>

Znaczenie: próba wykonania operacji dzielenia przez zero. Wynik operacji jest równy zeru.

Reakcja programu: wykonywanie programu jest kontynuowane.

NADMIAR URZĄDZENIA x

x -<symbol-urządzenia>

Znaczenie: próba wykonania instrukcji ACCEPT lub READ (dla zbioru związanego z symbolicznym wejściem-wyjściem), gdy wystąpił sygnał końca danych na urządzeniu <symbol-urządzenia>.

Reakcja programu: przerwanie wykonywania programu.

NIEODCZYTYWALNY BLOK ZBIORU x y z

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

z -<numer-bloku>

Znaczenie: żadne z powtórzeń zapisu bloku o numerze <numer-bloku> nie daje się odczytać.

Reakcja programu: przerwanie wykonywania programu.

NIE OTWARTO ZBIORU x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

Znaczenie: nie można wykonać instrukcji OPEN, ponieważ

a) na szpuli nie ma metryki początku szpuli albo metryka ta jest niepoprawna, albo niedostępna,

b) stacja <symbol-urządzenia> została odłączona od zestawu,

c) stacja <symbol-urządzenia> jest fizycznie zablokowana dla zapisu (dla OPEN OUTPUT),

d) na taśmie znajduje się źle zapisany blok.

Reakcja programu: następuje zawieszenie maszyny. Po jej odwieszeniu przez operatora instrukcja OPEN wykonywana jest ponownie.

NIE OTWARTY ZBIOR x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

- Znaczenie: a) próba wykonania instrukcji READ lub WRITE dla nieotwartego zbioru,
- b) próba wykonania instrukcji READ, gdy zbiór jest otwarty do pisania,
- c) próba wykonania instrukcji WRITE, gdy zbiór jest otwarty do czytania.

Reakcja programu: przerwanie wykonywania programu.

NIEPODPRAWNE OTWARCIE ZBIORU x y

x -<identyfikator-zbioru>
y -<symbol-urządzenia>

- Znaczenie: a) próba otwarcia otwartego zbioru,
- b) próba otwarcia zbioru mającego wspólny bufor (SAME AREA) lub wspólne urządzenie ze zbiorem już otwartym,
- c) próba otwarcia zbioru na szpulii wielozbiorowej, na której jest już otwarty inny zbiór,
- d) próba otwarcia zbioru, dla którego została w programie wykonana instrukcja CLOSE WITH LOCK.

Reakcja programu: przerwanie wykonywania programu.

NIE ROZPOZNANY DOKUMENT ZBIORU x y z

x -<identyfikator-zbioru>
y -<symbol-urządzenia>
z -<numer-kolejny-dokumentu>

Znaczenie: monit dotyczy czytania zbiorów związanych z symbolicznym wejściem i pojawia się gdy -

- a) długość dokumentu wczytanego ze zbioru pozycyjnego nie jest zgodna z żadnym spośród opisów dokumentów tego zbioru,
- b) liczba pozycji w dokumencie wczytanym ze zbioru niepozycyjnego techniką 1 nie jest zgodna z żadnym spośród opisów dokumentów tego zbioru,
- c) dokument wczytany ze zbioru niepozycyjnego techniką 2 ma niepoprawny znacznik dokumentu,
- d) nie można rozpoznać typu dokumentu wczytanego ze zbioru niepozycyjnego techniką 3.

We wszystkich omówionych przypadkach rejestr TALLY nie zmienia wartości. <numer-kolejny-dokumentu> jest liczbą całkowitą, która określa numer dokumentu względem początku zbioru.

Reakcja programu: wykonywanie programu jest kontynuowane.

NIE ZAMKNIĘTO ZBIORU x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

Znaczenie: próba wykonania instrukcji STOP RUN, gdy zbiór na taśmie magnetycznej nie jest zamknięty; w takim przypadku zbioru taśmowego otwartego dla wyjścia nie można przy powtórным wykorzystaniu odczytać do końca, gdyż

- a) ostatnia zawartość bufora zbioru może nie być zapisana,
- b) nie ma metryki końca zapisu na szpuli.

Reakcja programu: przerwanie wykonywania programu, po sprawdzeniu czy zostały zamknięte wszystkie zbiory danych.

OTWARTO ZBIOR x y

x - <identyfikator-zbioru>
y - <symbol-urządzenia>

Znaczenie: jest to sygnał zakończenia wszystkich czynności związanych z otwarciem zbioru.

Reakcja programu: wykonywanie programu jest kontynuowane.

Uwaga Omawiany monit wypisywany jest tylko wtedy, gdy program źródłowy poprzedzony jest dyrektywą MON.

PODAJ NUMER GENERACJI x y

x - <identyfikator-zbioru>
y - <symbol-urządzenia>

Znaczenie: monit ten pojawia się w trakcie otwarcia zbioru opisanego zwrotem LABEL RECORD STANDARD WITH GENER; operator maszyny musi na monitorze podać numer generacji zbioru, który ma postać liczby całkowitej pięciocyfrowej, zakończonej gwiazdką (np. 00012*).

Reakcja programu: wykonywanie programu jest kontynuowane.

PUSTE PRZESŁANIE ZBIORU x y z

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

z -<numer-bloku>

Znaczenie: nie można wykonać przesłania taśmowego bloku <numer-bloku> (instrukcje READ, WRITE, OPEN lub CLOSE) dla zbioru <identyfikator-zbioru>.

Reakcja programu: przerwanie wykonywania programu.

WYKŁADNIK POTĘGI UJEMNY LUB $0^{*}0$ x

x -<numer-segmentu>

Znaczenie: próba wykonania operacji potęgowania, gdy wykładnik potęgi jest ujemny albo próba wykonania operacji $0^{*}0$; wynik każdej z tych operacji jest równy zeru.

Reakcja programu: wykonywanie programu jest kontynuowane.

WYKŁADNIK PRZEKROCZYŁ ZAKRES x

x -<numer-segmentu>

Znaczenie: wynik jakiegoś działania zmiennoprzecinkowego wykonanego w segmencie <numer-segmentu> wykracza poza zakres liczby zmiennoprzecinkowej.

Reakcja programu: wykonywanie programu jest kontynuowane.

WYKONANO RERUN x y z

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

z -<numer-bloku>

Znaczenie: jest to sygnał zakończenia wszystkich czynności związanych z tworzeniem punktu ponawiania programu na zbiorze <identyfikator-zbioru>; blok RERUN ma w tym zbiorze numer <numer-bloku>.

Reakcja programu: wykonywanie programu jest kontynuowane.

WYNIK SORTOWANIA x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

Znaczenie: a) wynik działania instrukcji SORT z opcją GIVING (jeżeli zbiorowi przydzielono jedno z urządzeń zbioru sortowania) znajduje się na urządzeniu <symbol-urządzenia>

Reakcja programu: wykonanie programu jest kontynuowane.

ZA DŁUGI BLOK ZBIORU x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

Znaczenie: blok zbioru, czytany z taśmy magnetycznej, nie mieści się w buforze zbioru.

Reakcja programu: czytany blok jest stracony do długości równej buforowi zbioru; nie jest przy tym sprawdzana suma kontrolna bloku. Wykonywanie programu jest kontynuowane.

ZA DUŻA KROTNOSC PERFORM x

x --<numer-segmentu>

Znaczenie: próba wykonania instrukcji PERFORM zagnieżdżonej w ośmiu niezakończonych instrukcjach PERFORM.

Reakcja programu: przerwanie wykonywania programu.

ZA DUŻO BUFORÓW

Znaczenie: przekroczenie dopuszczalnej liczby buforów dla zbiorów, sygnalizowane w trakcie wykonywania instrukcji OPEN.

Reakcja programu: przerwanie wykonywania programu.

ZA DUŻY DOKUMENT ZBIORU x y

x --<identyfikator-zbioru>

y --<symbol-urządzenia>

Znaczenie: a) dokument czytany z taśmy magnetycznej nie mieści się w polu dokumentu,

b) dokument wyjściowego zbioru związanego z taśmą magnetyczną jest większy od bufora tego zbioru.

Reakcja programu: dla a) dokument jest obcinany do długości pola dokumentu; TALLY staje się równe długości nie obciętego dokumentu liczonej w słowach maszyny; wykonywanie programu jest kontynuowane,

dla b) wypisywany jest monit PRZYCZYNA ZAKOŃCZENIA 125 i następuje przerwanie wykonywania programu.

ZA KRÓTKIE POLE ZBIORU x y

x - <identyfikator-zbioru>

y - <symbol-urządzenia>

Znaczenie: dotyczy zbiorów związanych z taśmą magnetyczną; próba wykonania instrukcji OPEN INPUT, gdy długość bufora (lub pola dokumentu) podana w opisie zbioru w programie, jest mniejsza od długości bloku (lub dokumentu) podanej w metryce początku tego zbioru.

Reakcja programu: przerwanie wykonywania programu.

ZAMKNIĘCIE ZAMKNIĘTEGO ZBIORU x y

x - <identyfikator-zbioru>

y - <symbol-urządzenia>

Znaczenie: próba zamknięcia zamkniętego zbioru.

Reakcja programu: przerwanie wykonywania programu.

ZAMKNIĘTO ZBIÓR LUB SZPUŁĘ x y z

- x -<identyfikator-zbioru>
- y -<symbol-urządzenia>
- z -<ilość-błędnych-przesłań>

Znaczenie: monit ten pojawia się po wykonaniu wszystkich czynności związanych z zamknięciem zbioru <identyfikator-zbioru> lub jednej z jego szpul (dla zbiorów wieloszpułowych) na stacji taśmy magnetycznej <symbol-urządzenia>; liczba całkowita <ilość-błędnych-przesłań> oznacza liczbę błędów powstałych w czasie czytania taśmy (lub pisania na taśmie) i poprawionych automatycznie; liczba ta charakteryzuje stopień zużycia nośnika.

Reakcja programu: wykonywanie programu jest kontynuowane.

Uwaga Omawiany monit wypisywany jest tylko wtedy, gdy program źródłowy jest poprzedzony dyrektywą MON.

ZAMKNIĘTO ZBIÓR x y

- x -<identyfikator-zbioru>
- y -<symbol-urządzenia>

Znaczenie: monit sygnalizuje zakończenie wszystkich czynności związanych z zamknięciem zbioru <identyfikator-zbioru>, związanego z symbolicznym wejściem lub wyjściem <nazwa-urządzenia>.

Reakcja programu: wykonywanie programu jest kontynuowane.

Uwaga Omawiany monit wypisywany jest tylko wtedy, gdy program źródłowy poprzedzony jest dyrektywą MON.

ZBIOR-GENERACJA-SZPULA x y z
ZAŁOŻ NA w

x - <identyfikator-zbioru>
y - <numer-generacji>
z - <numer-szpuli>
w - <symbol-urządzenia>

Znaczenie: zaczyna się wykonywanie programu od pewnego punktu ponawiania; w związku z tym należy zbiór taśmowy określony parametrami <identyfikator-zbioru>, <numer-generacji>, <numer-szpuli> założyć na stacji taśmy magnetycznej <symbol-urządzenia>.

Reakcja programu: następuje zawieszenie maszyny; po założeniu szpuli zbioru na zadanej stacji operator winien odwieść maszynę.

ZBIOR-ZEROWATOR x y
ZAŁOŻ NA z

x - <identyfikator-zbioru>
y - <numer-zerowatora>
z - <symbol-urządzenia>

Znaczenie: zaczyna się wykonywanie programu od pewnego punktu ponawiania; w związku z tym należy zbiór <identyfikator-zbioru>

założyć na symboliczne wejście <symbol-urządzenia> możliwie blisko przed zerowatorem <numer-zerowatora>.

Reakcja programu: następuje zawieszenie maszyny; po założeniu szpuli zbioru na zadanej stacji operator winien odwiesić maszynę.

ZLA KOLEJNOSC DAT x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

Znaczenie: monit ten dotyczy wielozbiorowej szpuli taśmy magnetycznej i wykonania instrukcji OPEN OUTPUT, gdy data ważności otwieranego zbioru <identyfikator-zbioru> jest większa od daty ostatniego aktualnego zbioru na szpuli.

Reakcja programu: przerwanie wykonywania programu.

ZLA ORGANIZACJA DANYCH ZBIORU x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

Znaczenie: monit ten dotyczy zbioru związanego z taśmą magnetyczną; występuje wtedy, gdy

- a) zapis na taśmie nie jest zgodny ze standardem przyjętym w języku COBOL (dotyczy czytania oraz otwarcia na wejściu albo wyjściu),

b) metryki na szpuli wielozbiorowej występują w złej kolejności.

Reakcja programu: przerwanie wykonywania programu.

ZŁA SUMA KONTROLNA ZBIORU x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

Znaczenie: suma kontrolna zbioru, obliczona podczas jego zapisywania i umieszczona w metryce końca tego zbioru, nie zgadza się z sumą kontrolną obliczoną w trakcie czytania tego zbioru.

Reakcja programu: przerwanie wykonywania programu.

ZMIEN SZPULE x y

x -<identyfikator-zbioru>

y -<symbol-urządzenia>

Znaczenie: monit ten dotyczy zbiorów wieloszpułowych związanych z jedną stacją taśmy magnetycznej; pojawia się on po zamknięciu szpuli zbioru <identyfikator-zbioru>; na stację taśmy magnetycznej <symbol-urządzenia> należy założyć kolejną szpulę zbioru.

Reakcja programu: po wypisaniu monitu następuje zawieszenie maszyny; operator powinien założyć nową szpulę taśmy, a następnie odwiesić maszynę; wtedy wykonywanie programu jest kontynuowane.

MONITY TRANSLATORA ZAKOŃCZEŃ

=====

Translator Zakończeń wywoływany jest przez System Operacyjny, gdy w czołówce w zdaniu PROBLEM, następną instrukcją po instrukcji PROGRAM jest ZCOB.

Podajemy alfabetyczny spis monitów Translatora Zakończeń. Monity te są wypisywane na zerowym wyjściu programu.

BŁĘDNE ODWOŁANIE DO BIBLIOTEKI SYSTEMU

Znaczenie: próba wykonania rozkazu odwołującego się do zbioru lub podzbioru z Biblioteki Systemu jeśli

- a) w Bibliotece nie ma takiego zbioru lub podzbioru,
- b) zbiór przekracza obszar bębnowy,
- c) sekcja startowa zbioru lub podzbioru przekracza obszar PAO.

BŁĘDNE PARAMETRY OTWORZ UZ

Znaczenie: próba wykonania rozkazu OTWORZ UZ jeśli na skutek błędnego parametru, tj. adresu metryki PAO - obszar przeznaczony na metrykę wykracza poza PAO programu.

BŁĘDNE PARAMETRY SDY CZYTAJ MON

Znaczenie: błędnie określony adres pola powoduje, że kolejny znak czytany z monitora nie mieści się w PAO.

BŁĘDNE PARAMETRY SDY DEFINIUJ LUB WYMIEN PROGRAM NR URZĄDZENIA = x y z

x - <liczba>

y - <liczba>

z - <liczba>

Znaczenie: próba wykonania rozkazu DEFINIUJ PRZESŁANIE lub WYMIEN PROGRAM gdy zachodzi jedna z sytuacji

- a) podany numer urządzenia dotyczy nie istniejącego w zestawie urządzenia,
- b) 'apao' + 'ilość' przekracza zakres pamięci operacyjnej,
- c) 'ilość' jest liczbą ujemną lub zerem.

BYŁY BŁĘDY FORMALNE W PROGRAMIE

Znaczenie: podczas tłumaczenia programu translator wykrył w programie błędy formalne.

BYŁY BŁĘDY PODCZAS WYKONANIA PROGRAMU WYNIKOWEGO

Znaczenie: monit spowodowany jest jedną z następujących przyczyn

- a) podczas wykonania programu roboczego powstały błędne sytuacje, o których translator sygnalizował wypisanie na monitorze,
- b) wykonanie rozkazu - STO 125 - zawarte-go we wstawce SAS.

INICJACJA NIEZDEFINIOWANEGO PRZESŁANIA

Znaczenie: próba wykonania rozkazu INICJUU PRZESŁANIE, podczas gdy przesłanie to nie było uprzednio zdefiniowane.

KONIEC WYKONANIA PROGRAMU ROBOCZEGO

Znaczenie: koniec działania programu roboczego spowodowany wykonaniem instrukcji STOP RUN lub rozkazu STO 15, zawartego we wstawce SAS.

NIELEGALNY ROZKAZ LUB SDY NIETYPOWE Z BŁĘDNYM PARAMETREM W CZĘŚCI ADRESOWEJ

Znaczenie: monit ten jest sygnalizowany z następujących przyczyn

- a) nielegalna część operacyjna,
- b) za duża krotność pośredniego adresowania,
- c) SDY NIETYPOWE z błędną częścią adresową.

NIEPRAWIDŁOWY ADRES STOPU

Znaczenie: koniec wykonania programu roboczego spowodowany nieokreśloną przyczyną lub wykonaniem rozkazu STO, ze stawki SAS, z częścią adresową większą od 399.

NIE ZDEFINIOWANY W CZOŁÓWCE DEKODER

Znaczenie: próba wykonania rozkazu P31 lub instrukcji czytania/pisania dotyczących wejścia/wyjścia, dla którego w czołówce nie został zdefiniowany dekodery.

ODWOLANIE DO TMS GDY ZWOLNIONA

Znaczenie: problem odwołuje się do zbiorów z Biblioteki Systemu, które nie zostały wyadaptowane na Bęben Systemu, podczas gdy taśma systemu została zwolniona.

PROBA DOLACZENIA NIE ZDEFINIOWANEGO W CZOŁOWCE WEJSCIA/WYJŚCIA

Znaczenie: próba dołączenia wejścia/wyjścia nie zdefiniowanego w czołówce.

PROBA DOLACZENIA WEJSCIA/WYJSCIA, GDY WYCZER- PAŁO WOLNE NUMERY PRZESŁAŃ

Znaczenie: próba dołączenia wejścia/wyjścia, gdy liczba wolnych numerów przesłań jest mniejsza od liczby przesłań wymaganych przez to wejście/wyjście.

PROBA DOLACZENIA WEJSCIA/WYJSCIA Z BŁEDNIE ZDEFINIOWANYM PODPROGRAMEM LUB DEKODEREM

Znaczenie: próba dołączenia wejścia/wyjścia, w którego definicji umieszczono nazwę nie-
dozwolonego podprogramu lub dekodera.

PROBA DOLACZENIA WEJSCIA/WYJSCIA, GDY PODPRO- GRAM LUB DEKODER PRZEKRACZA GÓRNĄ GRANICĘ PAO PAO [38] = x

x - <liczba>

Znaczenie: próba dołączenia wejścia/wyjścia, gdy zawartość PAO [38] równa sumie łącznej

długości podprogramu z dekoderm i poprzedniej zawartości PAO [38] przekracza górną granicę pamięci operacyjnej.

PROBA ROZDEFINIOWANIA PRZESŁANIA W TOKU

Znaczenie: próba wykonania rozkazu ROZDEFINIUJ PRZESŁANIE dotyczącego nie zakończonego przesłania.

PROBA SKORZYSTANIA Z NIE DOŁĄCZONEGO WEJSCIA/WYJSCIA

Znaczenie: próba wykonania rozkazu P31 lub instrukcji czytania/pisania dotyczących nie dołączonego wejścia/wyjścia.

PROBA SKORZYSTANIA Z WEJSCIA/WYJSCIA O NIEWŁAŚCIWIE ZDEFINIOWANYM DEKODERZE

Znaczenie: próba wykonania rozkazu P31 lub instrukcji czytania/pisania, gdy urządzenie związane z symbolicznym wejście/wyjściem nie może współpracować z dekoderm tego wejścia/wyjścia, np. dekoderm IM2P, a czytelnik nastawiony na czytanie taśmy 8 kanałowej.

PROBA WYKONANIA ROZKAZU P06, P07, ..., P11

Znaczenie: na skutek błędu logicznego w programie sterowanie przechodzi do miejsca pamięci, zawierającego jeden z rozkazów P06, P07, ..., P11.

PROBA WYKONANIA ROZKAZU PROGRAMOWANEGO PRZY
NIE USTAWIONEJ ZAWARTOSCI PAO [16], PAO [17],
..., PAO [30]

Znaczenie: próba wykonania rozkazu progra-
mowanego przy nie ustawionej zawartości
PAO [16], ..., PAO [30].

PROGRAM WYRZUCONY PRZEZ OPERATORA

Znaczenie: wyrzucenie programu przez ope-
ratora kluczem WYRZUĆ PROGRAM.

ZŁE DZIAŁANIE TRANSLATORA

Znaczenie: złe działanie translatora spowo-
dowane błędem w translatorze, w Systemie
Operacyjnym lub w maszynie.

PRZYCZYNA ZAKOŃCZENIA x

$x = 0, 1, \dots, 14, 16, 17, \dots, 124, 126, 127, \dots, 399$

Znaczenie: koniec wykonania programu wyniko-
wego; ostatnim wykonanym rozkazem było STO x,
zawarty we wstawce SAS.

PRZYCZYNA ZAKOŃCZENIA 15

Znaczenie: koniec wykonania programu wyniko-
wego; ostatnim wykonanym rozkazem był STO 15
lub instrukcja STOP RUN.

PRZYCZYNA ZAKOŃCZENIA 125

Znaczenie: program wynikowy zakończył dzia-
łanie na skutek błędów wykrytych podczas
wykonania programu lub po wykonaniu rozkazu
STO 125.

URZĄDZENIE NIE JEST TAŚMĄ MAGNETYCZNĄ

Znaczenie: próba wykonania rozkazu SDY, który dotyczy taśm magnetycznych, podczas gdy parametr UZ nie jest taśmą magnetyczną lub gdy ten rozkaz SDY jest zarezerwowany dla taśmy systemu.

ZAPETLENIE ADRES ADRESU W SDY

Znaczenie: za duża krotność pośredniego adresowania przy określeniu parametrów rozkazu SDY.

PRZYKŁADY PROGRAMÓW W JĘZYKU COBOL

Przykład 1

Jako przykład ilustrujący zastosowanie języka COBOL przyjęto system 2 programów realizujących automatyczną kontrolę stanu magazynu. Zagadnienie zostało uproszczone aby programów zbyt nie komplikować.

System składa się z Programu Tworzenia Ewidencji oraz Programu Głównego.

Program Tworzenia Ewidencji używany jest do zakładania ewidencji materiałowej na taśmie magnetycznej, mianowicie program ten przepisuje dokumenty inwentaryzacyjne z taśmy papierowej na taśmę magnetyczną sortując je jednocześnie wg nazwy artykułu (w porządku rosnącym). Właściwa część proceduralna tego programu składa się z jednej instrukcji, tj. instrukcji SORT i w związku z tym napisanie go nie wymagało przygotowania sieci działań.

Program Główny aktualizuje ewidencję zapasów na podstawie dokumentów rozchodowych i przychodowych oraz wypisuje zamówienia na artykuły, których ilość spadła poniżej stanu alarmowego. Zamówienia - sortowane wg nazw dostawców (w porządku rosnącym) - są wypisywane na drukarce wierszowej. Na specjalne żądanie (przez ustawienie wskaźnika) Program Główny wypisuje aktualny stan ewidencji materiałowej na drukarce wierszowej. Sieć działań części proceduralnej Programu Głównego przedstawiono w pkt 4 niniejszego Dodatku.

W sieci działań pominięto część DECLARATIVES i wstawkę dołączającą wyjście 1.

W systemie automatycznej kontroli zapasów w magazynie wprowadzono trzy rodzaje dokumentów: dokumenty ewidencji materiałowej, dokumenty rozchodowe i przychodowe czyli kwity oraz dokumenty związane z wydawnictwem stanu ewidencji i zamówień na drukarce wierszowej.

Dokumenty ewidencji bez względu na to, czy występują w zbiorze związanym z nośnikiem papierowym, czy taśmą magnetyczną i bez względu na nazwę mają w obu programach taką samą budowę tzn. składają się z czterech pozycji: z nazwy artykułu (pozycja tekstowa), nazwy dostawcy tego artykułu (pozycja tekstowa) i dwóch pozycji liczbowych obliczeniowych (typu FIXED), liczby sztuk w magazynie oraz stanu alarmowego.

Kwity rozchodowe i przychodowe mają taką samą budowę. Składają się z trzech pozycji: ze wskaźnika, który przybiera wartość 'KWP' dla kwitu przychodowego i 'KWR' dla rozchodowego, nazwy artykułu (pozycja tekstowa) i ilości (wydanej gdy 'KWR' lub przyjętej do magazynu gdy 'KWP'). Kwity używane są w Programie Głównym do aktualizacji EWIDENCJI. Kwity muszą być ułożone w kolejności rosnących nazw artykułów. Zła kolejność kwitów powoduje w Programie Głównym wypisanie komunikatu i przerwanie pracy. Kwity przed wprowadzeniem do Programu Głównego można oczywiście posortować programem, który byłby analogiczny do Programu Tworzenia Ewidencji i wobec tego został pominięty w opisie przykładu.

Dokument wydawniczy zwany w programie WYDRUK ma dwie pozycje: nazwę artykułu (pozycja tekstowa) i ilość (pozycja liczbowa-wydawnicza). Dokument WYDRUK jest używany do wypisywania stanu ewidencji materiałowej na dzień bieżący oraz do wypisywania zamówień na artykuły, których ilość w magazynie spadła poniżej 'Stanu alarmowego'. Przyjęto, że zamawia się ilość równą potrojonemu stanowi alarmowemu.

W przykładzie posłużono się fikcyjnymi danymi dotyczącymi magazynu mebli.

W obu programach, prócz właściwej części proceduralnej, jest część DECLARATIVES, w której za pomocą zmiennej BUFF są wypisywane błędne dokumenty wejściowe.

W dalszym ciągu dodatku podane są następujące materiały ilustrujące opis programów:

- 1) tekst źródłowy Programu Tworzenia Ewidencji,
- 2) tabulogram taśmy papierowej zawierającej dane dla Programu Tworzenia Ewidencji oraz monity czasu wykonania,
- 3) wypis błędnego dokumentu,
- 4) sieć działań Programu Głównego,
- 5) tekst źródłowy Programu Głównego,
- 6) tabulogram taśmy papierowej zawierającej dane dla Programu Głównego,
- 7) stan ewidencji - drukowany przez Program Główny na drukarce wierszowej,
- 8) zamówienia dla dostawców - drukowane przez Program Główny na drukarce wierszowej.

9) Wydruk z monitora zawierający:

monity czasu wykonania ilustrujące pracę Programu Głównego (wypisywane na skutek umieszczenia w czołówce języka COBOL Programu Głównego dyrektywy MON) oraz komunikaty instrukcji STOP 'tekst' i odpowiedzi operatora.

1. TEKST ŹRÓDŁOWY PROGRAMU TWORZENIA EWIDENCJI

WYT:O=PDW(DX,DDX1);
WEP:O=PCTP(CTP1,DM28);
WYP:O=PDW(DW,DDW8);
TYTUL:PROGRAM TWORZENIA EWIDENCJI;
PROBLEM:COBO,PROG.

MON;TAB.

IDENTIFICATION DIVISION.
PROGRAM..ID.PROGRAM TWORZENIA EWIDENCJI.

ENVIRONMENT DIVISION.

INPUT..OUTPUT SECTION.
FILE..CONTROL.

SELECT EWIDENCJA..PAPIEROWA ASSIGN TO INPUT O.
SELECT EWIDENCJA ASSIGN TO MTAPE A, MTAPE B
OR MTAPE C.
SELECT ZBIOR..SORTOWANIA ASSIGN TO MTAPE A,
MTAPE B, MTAPE C.

DATA DIVISION.

FILE SECTION.

FD EWIDENCJA..PAPIEROWA; LABEL RECORD OMITTED;
DATA RECORD IS KARTAP.
01 KARTAP.
02 NAZWA; PICTURE IS 'X(20)'.
02 DOSTAWCA; PICTURE IS 'X(45)'.
02 ILOSC; FIXED.
02 STAN..ALARMOWY; FIXED.

FD EWIDENCJA; LABEL RECORD IS STANDARD;
VALUE OF IDENT IS 'EWID';
DATA RECORD IS KARTA.
01 KARTA.
02 NAZWA; PICTURE IS 'X(20)'.
02 DOSTAWCA; PICTURE IS 'X(45)'.
02 ILOSC; FIXED.
02 STAN..ALARMOWY; FIXED.

SD ZBIOR..SORTOWANIA;BLOCK 3 RECORDS;
DATA RECORD IS KARTAS.
01 KARTAS.
02 NAZWA;PICTURE IS 'X(20)'.
02 DOSTAWCA; PICTURE IS 'X(45)'.
02 ILOSC; FIXED.
02 STAN..ALARMOWY; FIXED.

PROCEDURE DIVISION.

DECLARATIVES.
B SECTION.

USE AFTER STANDARD ERROR PROCEDURE ON INPUT.

B1. DISPLAY '.LF',BUFF OF EWIDENCJA..PAPIEROWA,
' .LF' UPON OUTPUT 0.

END DECLARATIVES.

S SECTION.

SORT ZBIOR..SORTOWANIA ON ASCENDING KEY
NAZWA IN KARTAS
USING EWIDENCJA..PAPIEROWA
GIVING EWIDENCJA.
STOP RUN.

END PROGRAM.

2. TABULOGRAM TAŚMY PAPIEROWEJ ZAWIERAJĄCEJ

DANE DLA PROGRAMU TWORZENIA EWIDENCJI

fotel-wyscielany wphm-warszawa-nowogrodzka-3 50 20
 krzeslo-wyscielane wphm-wroclaw-odrzanska-10 1500 450
 krzeslo-twarde wfm-wyszkow-stoleczna-7 5850 1500
 tapczan-tapicerski wphm-warszawa-nowogrodzka-3 800 300
 biurko-orzech wfm-wyszkow-stoleczna-7 1200 400
 biurko-gabinetowe spoldzielnia -antyk-warszawa-plocka-6 20 5
 fotel-bujany rekodzielo-artyst-poznan-za-grobla-3 50 10
 wersalka wphm-krakow-sw-wita-6 500 150
 regal-sosna cplia-warszawa-poznanska-3 560 300
 regal-palony rekodzielo-artyst-poznan-za-grobla-3 356 140
 szafka-kuch-kss9 pfm-swarzedz-warszawska 150 2825 1500
 szafka-kuch-agata pfm-swarzedz-warszawska-150 5652 2500
 stol-kuch-ludwik pfm-swarzedz-warszawska-150 3867 1500
 stolik-pod-tv spoldz-postep -klaj -prusa-6 628 200
 komplet-karlik jugmebex-belgrad-312 16 5
 fotel-s114 dmv-lipsk-618 88 20
 =====

Monity czasu wykonania

O:WOLNA 11 1' 50''
 O:OTWARTO ZBIOR ROBOCZY SORT-2 B
 O:OTWARTO ZBIOR ROBOCZY SORT-1 A
 O:OTWARTO ZBIOR EWIDENCJA..PAPIERO O
 O:ZAMKNIETO ZBIOR EWIDENCJA..PAPIERO O
 O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-2 B O
 O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-1 A O
 O:OTWARTO ZBIOR ROBOCZY SORT-2 B
 O:OTWARTO ZBIOR ROBOCZY SORT-1 A
 O:OTWARTO ZBIOR ROBOCZY SORT-3 C
 O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-3 C O
 O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-1 A O
 O:OTWARTO ZBIOR ROBOCZY SORT-3 C
 O:WYNIK SORTOWANIA EWID A
 O:OTWARTO ZBIOR EWID A
 O:ZAMKNIETO ZBIOR LUB SZPULE EWID A O
 O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-3 C O
 O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-2 B O
 O:PROBL 11 9' 59''

3. WYPIS BŁĘDNYCH DOKUMENTÓW

ZAM 41 - SO 141 BCN(B) 255/032 - ZEST 0(C) - 25.10.71(298)/1
SUPERVISOR = SVC PAC = 15672 DEB = 42384

BIURKO-GABINETOWE SPOLDZIÉLNIA -ANTYK-WARSZAWA-PŁOCKA-6 20 3
SZAFKA-KUCH-KSS9 PFM-SWARZEDZ-WARSZAWSKA 150 2825 1500
STOLIK-POD-TV SPOLDZ-POSTEP -KLAJ -PRUSA-6 628 200

5. TEKST ŹRÓDŁOWY PROGRAMU GŁÓWNEGO

```
WYT:O=PDW(DX,DDX1);
WEP:O=PCTP(CTP1,DM28);
WYP:O=PDW(DW,DDW8),
      1=PPTP(PTP1,DM2);
TYTUL:PROGRAM GŁÓWNY;
PROBLEM:COBO, PROG.
```

MON ;TAB.

IDENTIFICATION DIVISION.
PROGRAM..ID. PROGRAM GŁÓWNY.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.
SPECIAL..NAMES.
SWITCH 1 ON IS DRUK.

INPUT..OUTPUT SECTION.
FILE..CONTROL.

```
SELECT EWIDENCJA ASSIGN TO MTAPE A.
SELECT NOWA..EWIDENCJA ASSIGN TO MTAPE B.
SELECT ZAMOWIENIA ASSIGN TO MTAPE C.
SELECT SORTOWANIE ASSIGN TO MTAPE A,
      MTAPE B, MTAPE C.
SELECT DRUKOWANIE ASSIGN TO OUTPUT O.
SELECT KWITY ASSIGN TO INPUT O.
```

DATA DIVISION.
FILE SECTION.

```
FD EWIDENCJA; LABEL RECORD IS STANDARD;
VALUE OF IDENT IS 'EWID';
DATA RECORD IS KARTA.
```

```
01 KARTA.
02 NAZWA; PICTURE IS 'X(20)'.
02 DOSTAWCA; PICTURE IS 'X(45)'.
02 ILOSC; FIXED.
02 STAN..ALARMOWY; FIXED.
```

```
FD NOWA..EWIDENCJA; LABEL RECORD IS STANDARD;
VALUE OF IDENT IS 'EWID';
DATA RECORD IS NOWA..KARTA.
```

01 NOWA..KARTA.

02 NAZWA; PICTURE IS 'X(20)'.

02 DOSTAWCA; PICTURE IS 'X(45)'.

02 ILOSC; FIXED.

02 STAN..ALARMOWY; FIXED.

FD KWITY; LABEL RECORD IS OMITTED;
DATA RECORD IS KWIT.

01 KWIT.

02 WSK; PICTURE 'X(3)'.

02 NAZWA; PICTURE IS 'X(20)'.

02 ILOSC; FIXED.

FD ZAMOWIENIA; LABEL RECORD IS OMITTED;
DATA RECORD IS POZYCJA.

01 POZYCJA.

02 NAZWA; PICTURE IS 'X(20)'.

02 DOSTAWCA; PICTURE IS 'X(45)'.

02 ILOSC; FIXED.

FD DRUKOWANIE; LABEL RECORD IS OMITTED;
DATA RECORD IS WYDRUK.

01 WYDRUK.

02 NAZ; PICTURE IS 'X(20)'.

02 IL; PICTURE IS 'Z(8)9'.

SD SORTOWANIE; BLOCK 3 RECORDS;
DATA RECORD IS SORTKARTA.

01 SORTKARTA.

02 N; PICTURE IS 'X(20)'.

02 D; PICTURE IS 'X(45)'.

02 S; FIXED.

WORKING..STORAGE SECTION.

77 DATA..WYDRUKU; PICTURE 'X(25)'.

77 WSKZAM; FIXED; VALUE 0.

77 PODTYTUL; PICTURE 'X(45)'; VALUE '.45SP'.

PROCEDURE DIVISION.

DECLARATIVES.
BLAD SECTION 1.

USE AFTER STANDARD ERROR PROCEDURE ON KWITY.
B1.

DISPLAY '.LF', BUFF OF KWITY, '.LF' OUTPUT 1.

END DECLARATIVES.

ALFA SECTION 1.

A1. ENTER SAS;

SKO A

B) REZE 480

A) UEB B

PMB 38

SDY 1/11+1/13+1/17+8

ENTER COBOL.

OPEN INPUT EWIDENCJA, KWITY.

OPEN OUTPUT NOWA..EWIDENCJA, ZAMOWIENIA.

STOP 'USTAW.SPWSKAZNIK'.

IF Druk THEN NEXT ELSE GO TO CZYD.

ACCEPT DATA..WYDRUKU.

DISPLAY (5)'.LF', (5)'.SP', 'STAN.SPEWIDENCJI
SPNA.SPDZIEN.LF',

(10)'.SP', DATA..WYDRUKU, (2)'.LF'.

DISPLAY (4)'.SP', 'NAZWA', (15)'.SP', 'ILOSC.LF'.

OPEN OUTPUT DRUKOWANIE.

BETA SECTION 60.

CZYD.

READ KWITY AT END GO TO KONI .

CZYEW.

READ EWIDENCJA AT END GO TO KON .

POROWNANIE.

IF NAZWA IN EWIDENCJA EQUAL NAZWA IN KWIT
THEN GO TO ZALATW ELSE NEXT.

IF NAZWA IN EWIDENCJA LESS NAZWA IN KWIT
THEN GO TO PISZ ELSE NEXT.

DISPLAY 'ZLA .SP KOLEJNOSC.SP KWITOW.SP',
NAZWA IN KARTA,NAZWA IN KWIT OUTPUT 1.
STOP RUN.
PISZ.
MOVE KARTA TO NOWA..KARTA.
WRITE NOWA..KARTA.
IF ILOSC IN KARTA LESS STAN..ALARMOWY IN
KARTA THEN GO TO ZAMOW ELSE GO TO PISANIE.
ZALATW.
IF WSK EQUAL 'KWP' THEN GO TO PRZYCHOD ELSE NEXT.
ROZCHOD.
COMP ILOSC IN KARTA = ILOSC IN KARTA - ILOSC IN KWIT.
GO TO CZYT.
PRZYCHOD.
COMP ILOSC IN KARTA=ILOSC IN KARTA + ILOSC IN KWIT.
CZYT.
READ KWITY AT END GO TO KON1.
GO TO POROWNANIE.
ZAMOW.
COMPUTE WSKZAM=WSKZAM + 1.
COMP ILOSC IN POZYCJA=3* STAN..ALARMOWY IN KARTA .
MOVE DOSTAWCA IN KARTA TO DOSTAWCA IN POZYCJA.
MOVE NAZWA IN KARTA TO NAZWA IN POZYCJA.
WRITE POZYCJA.
ZAM1. EXIT.
ZAM2.
GO TO PISANIE.
KON1.
IF ILOSC IN KARTA LESS STAN..ALARMOWY IN KARTA
THEN PERFORM ZAMOW THRU ZAM1 ELSE NEXT.
KON1.
MOVE KARTA TO NOWA..KARTA.
WRITE NOWA..KARTA.
IF DRUK THEN PERFORM PER THRU PER1 ELSE NEXT.
READ EWIDENCJA AT END GO TO KON .
GO TO KON1.
PISANIE.
IF DRUK THEN NEXT ELSE GO TO CZYEW.
PER.
MOVE NAZWA IN KARTA TO NAZ.
MOVE ILOSC IN KARTA TO IL.
WRITE WYDRUK. DISPLAY '.LF'.
PER1. EXIT.
PER2.
GO TO CZYEW.

GAMA SECTION 70.

KON.

CLOSE EWIDENCJA, NOWA..EWIDENCJA, ZAMOWIENIA, KWITY.
IF DRUK THEN CLOSE DRUKOWANIE ELSE NEXT.
IF WSKZAM EQUAL 0 THEN GO TO ST1 ELSE NEXT.
OPEN OUTPUT DRUKOWANIE.

STOP 'PROSZE.SP ZMIENIC.SP MT.SP A.SP.B.LF'.
DISPLAY (5)'.LF',(5)'.SP','ZAMOWIENIA.SP
DLA.SP DOSTAWCOW.LF'.
SORT SORTOWANIE ON ASCENDING KEY D
USING ZAMOWIENIA
OUTPUT PROCEDURE IS WYPIS.
CLOSE DRUKOWANIE.

ST1. STOP RUN.

WYPIS SECTION 70.

W1:

RETURN SORTOWANIE AT END GO TO KOPROG.
MOVE S TO IL.
MOVE N TO MAZ.
IF PODTYTUL IS LESS D THEN NEXT ELSE GO TO W2.
MOVE D TO PODTYTUL.
DISPLAY '.LF',PODTYTUL,'.LF',(45)'.MI','.LF'.

W2.

WRITE WYDRUK.DISPLAY '.LF'.
GO TO W1.

KOPROG. EXIT.

END PROGRAM.

6. DANE DO PROGRAMU GŁÓWNEGO

'1PAZDZIERNIKA 1971R';
KWP BIURKO-ORZECH 5
KWR BIURKO-ORZECH 908
KWR FOTEL-BUJANY 45
KWR KRZESLO-TWARDE 4500
KWR REGAL-SOSNA 300
KWR STOL-KUCH-LUDWIK 2500
KWR SZAFKA-KUCH-AGATA 3500

7. STAN EWIDENCJI

=====

STAN EWIDENCJI NA DZIEŃ
1 PAŹDZIERNIKA 1971R

NAZWA	ILOSC
BIURKO-ORZECH	297
FOTEL-BUJANY	5
FOTEL-S114	88
FOTEL-WYSCIELANY	50
KOMPLET-KARLIK	16
KRZESLO-TWARDE	1350
KRZESLO-WYSCIELANE	1500
REGAL-PALONY	356
REGAL-SOSNA	260
STOL-KUCH-LUDWIK	1367
SZAFKA-KUCH-AGATA	2152
TAPCZAN-TAPICERSKI	800
WERSALKA	500

=====

8. ZAMÓWIENIA DLA DOSTAWCÓW

ZAMOWIENIA DLA DOSTAWCÓW

CPLIA-WARSZAWA-POZNANSKA-3

REGAL-SOSNA 900

PFM-SWARZEDZ-WARSZAWSKA-150

SZAFKA-KUCH-AGATA 7500

STOL-KUCH-LUDWIK 4500

REKODZIELO-ARTYST-POZNAN-ZA-GROBLA-3

FOTEL-BUJANY 30

WFM-WYSZKOW-STOLECZNA-7

KRZESLO-TWARDE 4500

BIURKO-ORZECH 1200

9. MONITY CZASU WYKONANIA

=====

O:WOLNA 13	0' 16''		
O:OTWARTO ZBIOR EWID		A	
O:OTWARTO ZBIOR KWITY		O	
O:OTWARTO ZBIOR EWID		B	
O:OTWARTO ZBIOR ZAMOWIENIA		C	
O:USTAW WSKAZNIK			
O-W-1-1a			
O W= 010000000000000000000000			
O-Sa			
O S			
O:OTWARTO ZBIOR DRUKOWANIE		O	
O:ZAMKNIETO ZBIOR LUB SZPULE EWID			A
O:ZAMKNIETO ZBIOR LUB SZPULE EWID			B
O:ZAMKNIETO ZBIOR LUB SZPULE ZAMOWIENIA			C
O:ZAMKNIETO ZBIOR KWITY		O	
O:ZAMKNIETO ZBIOR DRUKOWANIE		O	
O:OTWARTO ZBIOR DRUKOWANIE		O	
O:PROSZE ZMIENIC MT A B			
O-Sa			
O S			
O:OTWARTO ZBIOR ROBOCZY SORT-2		B	
O:OTWARTO ZBIOR ROBOCZY SORT-1		A	
O:OTWARTO ZBIOR ZAMOWIENIA		C	
O:ZAMKNIETO ZBIOR LUB SZPULE ZAMOWIENIA			C
O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-2			B
O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-1			A
O:OTWARTO ZBIOR ROBOCZY SORT-2		B	
O:OTWARTO ZBIOR ROBOCZY SORT-1		A	
O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-2			B
O:ZAMKNIETO ZBIOR LUB SZPULE ROBOCZY SORT-1			A
O:ZAMKNIETO ZBIOR DRUKOWANIE		O	
O:PROBL 13	14' 54''		

Przykład 2

Przykład ten ilustruje użycie zwrotu RERUN i sposób ponawiania programu.

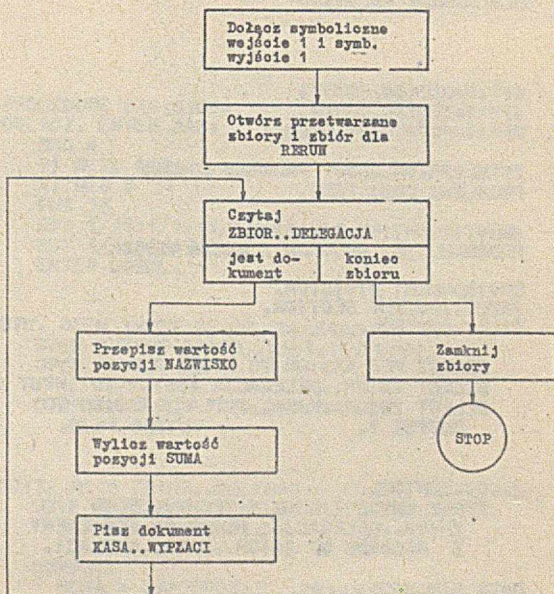
W programie tym na podstawie dokumentów "delegacje" wylicza się wysokość sumy, którą należy wypłacić delegowanemu pracownikowi.

W części ENVIRONMENT występuje żądanie wykonania akcji RERUN po każdym 5 wypisanych dokumentach "Kasa..wypłaci" (zob. załączony tabulogram przykładu).

W praktyce przy tak prostych wyliczeniach wykonania akcji RERUN będzie się żądać znacznie rzadziej (np. co kilkaset dokumentów).

Wyobraźmy sobie teraz, że po opracowaniu pewnej liczby dokumentów (większej niż 5) trzeba będzie przerwać wykonanie programu. Wówczas należy starannie przechować szpulę ze stacji A (przydzielonej zbiorowi, w którym są zapisane bloki rerunowe) oraz tekst wypisany na monitorze w czasie wykonania akcji RERUN.

SIEĆ DZIAŁAŃ



TABULOGRAM PROGRAMU

WYT:0-PDW(DW,DDW1);
WYP:1-PDW(DW,DDVB);
WEP:1-PCTP(CTP1,DN28);

TYTUL:PRZYKŁADOWY PROGRAM COBOL;
PROBLEM: COBO, PROG.

IDENTIFICATION DIVISION.
PROGRAM..ID. PROGRAM Z AKCJA RERUN.

ENVIRONMENT DIVISION.
INPUT..OUTPUT SECTION.
FILE..CONTROL.

SELECT RER ASSIGN TO MTAPE A.
SELECT ZBIOR..DELEGACJA ASSIGN TO INPUT 1.
SELECT ZBIOR..KASA..WYPLACI ASSIGN TO
OUTPUT 1.

1..0..CONTROL.
APPLY UNPOSITIONAL 1 TECHNIQUE ON
ZBIOR..DELEGACJA; RERUN ON RER EVERY
5 RECORDS OF ZBIOR..KASA..WYPLACI.

DATA DIVISION.
FILE SECTION.

FD RER; LABEL RECORD IS OMITTED;
DATA RECORD IS DOKRER.
01 DOKRER; PIC 'X(7)'.
FD ZBIOR..DELEGACJA; LABEL RECORD IS OMITTED;
BLOCK CONTAINS 50;
DATA RECORD IS DELEGACJA.
01 DELEGACJA.
03 NAZWISKO; PIC 'A(20)'.
03 MIEJSCOWOSC; PIC 'A(20)'.
03 DIETA ; FIXED.
03 HOTEL; FIXED.
03 IL..DNI; FIXED.
03 PODROZ; FIXED.
FD ZBIOR..KASA..WYPLACI; LABEL RECORD IS
OMITTED; DATA RECORD IS KASA..WYPLACI.
01 KASA..WYPLACI.
02 NAZWISKO; PIC 'A(20)'.
02 SUMA; PIC '9(6)'.

PROCEDURE DIVISION.

DOLACZ. ENTER SASI

SKO A

B) REZE 480

A) UEB B

PMB 38

SDY 1/11+1/13+1/17+8

SDY 1/11+1/17+8

ENTER COBOL.

OTW. OPEN INPUT ZBIOR..DELEGACJA.

OPEN OUTPUT ZBIOR..KASA..WYPLACI.

OPEN OUTPUT RER.

DISPLAY '.2LF KASA.SP WYPLACI

.2LF' OUTPUT 1.

CZYT. READ ZBIOR..DELEGACJA AT END GO TO KON.

MOVE NAZWISKO IN DELEGACJA TO NAZWISKO
IN KASA..WYPLACI.

COMPUTE

SUMA = 2*PODROZ+IL..DNI*(HOTEL+DIETA).

DISPLAY '.2LF' OUTPUT 1.

WRITE KASA..WYPLACI.

GO TO CZYT.

KON. CLOSE ZBIOR..DELEGACJA.

CLOSE ZBIOR..KASA..WYPLACI.

CLOSE RER.

STOP RUN.

END PROGRAM.

DANE DO PROGRAMU

==1

ADAMSKI WROCLAW 20 20 4 150
KOWALIK WROCLAW 20 20 4 150
STEFANSKI WROCLAW 25 35 2 200
BARANOWSKA GDANSK 20 35 2 108
SKARBK WARSZAWA 25 35 6 49
KULIGOWSKA-RATAJ WARSZAWA 25 35 4 49
ZWOLINSKI BYDGOSZCZ 20 20 5 83
RAMOWSKI BYDGOSZCZ 20 20 5 83
OKRASA TORUN 18 20 7 50
SADOWSKI TORUN 18 20 7 50
KIERSKA TORUN 18 25 7 50
JASINSKI TARDA 18 20 7 60

=====

WYNIK DZIAŁANIA PROGRAMU

ZAM 41 - SO 141 BC3(A) 326/633 - ZEST 0(C) - 3.11.71(307)/7
SUPERVISOR = SVC PAO = 15872 BEB = 42384

PRZYKŁADOWY PROGRAM COBOL
PROGRAM Z AKCJA RERUN.

LR=1578, PRZYCZYNA ZAKOŃCZENIA- 0
POZOSTAŁO 8388541 .CYK.

KASA WYPLACI

ADANSKI	.000460
KOWALIK	000460
STEFANSKI	000520
BARANOWSKA	000326
SKARBK	000458
KULIGOWSKA-RATAJ	000338
ZWOLINSKI	000366
RAMOWSKI	000366
OKRASA	000366
SADOWSKI	000366

Wydruk z monitora:

O:WOLNA 6 1' 14''
O:WYKONANO RERUN ZBIOR..KASA..WYPLA A 1
O:WYKONANO RERUN ZBIOR..KASA..WYPLA A 12
O:PROBL 6 5' 9''

Z powodu braku czasu przerywano wykonywanie programu (po wykonaniu drugiej akcji RERUN wypisano 10 dokumentów).

W celu kontynuowania wykonania programu należy zawiesić szpulę z blokami rerunowymi na dowolnej stacji np. na stacji B i spowodować wykonanie programu począwszy od pewnego (najlepiej ostatniego) punktu ponawiania. Ponawianie programu wykonuje się za pomocą translatora RESTART. Odwołanie do translatora RESTART opisane jest w pkt 5.2.

W opisanym powyżej przypadku wywołanie translatora RESTART będzie miało postać:

WY: O=PDW(DW,DDW1);
TYTUŁ: PONOWIENIE PROGRAMU PRZYKŁADOWEGO;
PROBLEM: REST, PROGRAM.

U=A, B=12.

Translator RESTART wypisze:

O: WOLNA 11 0' 39''
O: ZBIOR-ZEROWATOR ZBIOR..DELEGACJA 1
O: ZAŁOZ NA 1
O-SK
O S
O: PROBL 11 2' 54''

Po załozeniu danych do programu, na drukar-
ce otrzymamy dalszy ciąg dokumentów "Kasa..
wypłaci".

ZAM 41 - SO 141 RC3(A) 526/633 - ZEST 0(C). - 3.11.71(307)/14
SUPERVISOR = SVC .PAO = 15872 BEB = 42384

PONOWIENIE PROGRAMU PRZYKŁADOWEGO

KIERSKA 000401

JASINSKI 000386
####

LR=1936, PRZYCZYNA ZAKOŃCZENIA- 15
POZOSTAŁO 7999999 CYK.

WYDAWNICTWA IHM

Brzoźowy Ośrodek Informacji Naukowe-Technicznej i Ekonomicznej Instytutu Maszyn Matematycznych wydaje:

ALGORYTMY - półrocznik, zawiera artykuły na temat teorii programowania i zastosowania elektronicznych maszyn cyfrowych. Do nabycia w księgarni ORWN PAN oraz w Domach Książki. Cena zeszytu 40.- zł.

PRACE IHM - 3 numery w roku, zawierają publikacje naukowe i badawcze pracowników IHM w zakresie projektowania i budowy elektronicznych maszyn cyfrowych oraz systemów przetwarzania informacji. Do nabycia w księgarni ORWN PAN oraz w Domach Książki. Cena zeszytu 60.- zł.

Elektroniczna Technika Obliczeniowa - NOWOŚCI - kwartalnik, zawiera artykuły przeglądowe z dziedziny maszyn matematycznych, opracowane na podstawie najnowszej literatury światowej. Prenumeratę prowadzi Centrala Kolportażu Prasy i Wydawnictw "RUCH". Cena prenumeraty krajowej 240.- zł rocznie.

Automatyzacja Przetwarzania Informacji - INFORMACJA EKSPRESOWA - miesięcznik. Prenumeratę prowadzi Centrala Kolportażu Prasy i Wydawnictw "RUCH". Cena prenumeraty krajowej 240.- zł rocznie.