

P o l s k a A k a d e m i a N a u k
Z A K L A D A P A R A T O W M A T E M A T Y C Z N Y C H

prace
zakładu aparatów matematycznych p a n



Praca C. 2

SYSTEM AUTOMATYCZNEGO KODOWANIA
S A K O

Część I - Opis języka



Prace
ZAKŁADU APARATÓW MATEMATYCZNYCH
Polskiej Akademii Nauk

Praca C 2

SYSTEM AUTOMATYCZNEGO KODOWANIA
S A K O

Część I - Opis języka

Warszawa 1961

Copyright © 1961 - by Zakład Aparatów Matematycznych, Warszawa

Wszelkie prawa zastrzeżone

KOMITET REDAKCYJNY

Jerzy DAŃDA, Jerzy PIETT /z-ca redaktora/, Maria LESZEŻANKA
/sekr.redakcji/, Leon ŁUKASZEWICZ /redaktor/, Antoni MAZUR-
KIEWICZ, Tomasz PIETRZYKOWSKI, Zygmunt SAWICKI

Adres redakcji: Warszawa, ul. Koszykowa 79 tel. 21-84-41
wew. 131

System Automatycznego Kodowania SAKO opracowany został w latach 1959-1960 przez zespół następujących pracowników naukowych Zakładu Aparatów Matematycznych PAN

Leon Łukaszewicz
Antoni Mazurkiewicz

Jan Borowiec	Jerzy Swianiewicz
Jowita Koncewicz	Piotr Szorc
Maria Łacka	Alfred Szurman
Stefan Sawicki	Andrzej Wiśniewski

Ponadto w pracach nad SAKO udział wzięli:

Ludwik Czaja	Jacek Witaszek
Danuta Kosecka	Ewa Zaborowska

Podręcznik opracowali:

Leon Łukaszewicz
Antoni Mazurkiewicz

SAKO umożliwia automatyczne kodowanie programów, ułożonych dla maszyn XYZ oraz ZAM-2. Programy te, napisane w łatwym do opanowania języku SAKO, są następnie sprawdzane do postaci ostatecznej przez odpowiedni program kodujący, zapisany na stałe w pamięci bębnowej maszyny. W ten sposób sama maszyna wykonuje pracochłonne czynności kodowania, które w klasycznym programowaniu obciążały człowieka. Dzięki temu łączny czas przygotowania programu może być wielokrotnie skrócony. Sprawność programu wynikowego, uzyskanego przez program kodujący SAKO jest naogół taka, jak gdyby program ten wykonał sprawny programista.

SYSTEM AUTOMATYCZNEGO KODOWANIA
S A K O

Część I - Opis języka

SPIS RZECZY

WSTĘP

1. WSTĘPNE WIADOMOŚCI O MASZYNACH CYFROWYCH I PROGRAMOWANIU	7
.1. Schemat przetwarzania informacji	7
.2. Części składowe maszyn cyfrowych XYZ i ZAM-2	8
.1 Urządzenia wejściowe	8
.2 Urządzenia wyjściowe	10
.3 Pamięć	11
.4 Arytmometr	12
.5 Sterowanie	13
.3. Zapis dziesiętny liczb w maszynie	13
.1 Liczby ułamkowe	13
.2 Liczby całkowite	14
.4 Zapis binarny liczb w maszynie	15
.1 Liczby ułamkowa	15
.2 Liczby całkowite	17
.3 Słowo i działania bulowskie	17
.5. Ogólna struktura programów w języku SAKO	22
.6. Metodyka przygotowania programu	25
2. FORMA I OPIS POJĘĆ PODSTAWOWYCH W JĘZYKU SAKO	26
.1. Liczby	26
.1 Liczba ułamkowa	26
.2 Liczba całkowita	28
.3 Słowo bulowskie	29
.4 Blok liczbowy	29

.2. Zmienne	31
.1 Zmienna prosta	31
.2 Zmienna indeksowana	33
.3. Numery	34
.4. Funkcje	34
.5. Wyrażenia arytmetyczne	36
.6. Wyrażenia bulowskie	39
.7. Lista	41
3. FORMA I OPIS POSZCZEGÓLNYCH ROZKAZÓW SAKO	42
.1. Deklaracje	42
.2. Rozkazy sterujące	53
.3. Rozkazy arytmetyczne i bulowskie	63
.4. Rozkazy wejścia i wyjścia	69
.5. Rozkazy współpracy z bębnem	79
.6. Komentarz	81
4. PODPROGRAMY W JĘZYKU SAKO	81
.1. Terminologia i oznaczenia	82
.2. Ogólne właściwości systemu korzystania z podprogramów w języku SAKO	86
.3. Budowa podprogramów SAKO	87
.4. Korzystanie z podprogramów napisanych w Języku SAKO	89
SKOROWIDZ POJĘĆ PODSTAWOWYCH	97

W S T E P

System Automatycznego Kodowania - SAKO opracowany został celem znacznego zmniejszenia wysiłku i czasu, potrzebnych dla przygotowania programów. Jego zasadniczą rolą jest pełnienie funkcji tłumacza pomiędzy programistą a maszyną. Z jednej strony programista zapisuje program w formie podobnej do ogólnie przyjętej w matematyce, z drugiej strony maszyna otrzymuje ścisłe instrukcje dotyczące realizacji rozwiązania zadanego problemu. PROGRAM bowiem jest to jednoznaczny opis czynności maszyny, potrzebnych dla przetworzenia informacji wejściowych - d a n y c h w informacje wyjściowe - w y n i k i.

Dla rozwiązania na maszynie cyfrowej każdego problemu, należy wykonać dwa zadania:

- sformułować metodę rozwiązania problemu,
- zakodować tę metodę w języku zrozumiałym dla maszyny.

O ile pierwsze z tych zadań wymaga od programisty umiejętności i inteligencji w stopniu uwarunkowanym trudnością problemu, zadanie drugie wymaga tylko wiadomości o charakterze raczej formalnym i odpowiedniej wprawy w kodowaniu.

Programem zakodowanym w j e z y k u m a s z y n y nazywamy taki jego zapis, w którym wyszczególniony jest każdy rozkaz maszyny składający się na ten program. Przykładem takiego języka jest np. język SAS, opracowany dla maszyny XYZ i maszyny ZAM-2.

Z takim systemem kodowania związane są liczne trudności, a mianowicie:

- duża ilość czasu, potrzebna na wykonanie zadania i przekraczająca najczęściej czas potrzebny na sformułowanie metody rozwiązania,

- wynikający z powyższego długi okres czasu pomiędzy momentem postawienia zadania a jego rozwiązaniem,
- niedostateczna /znikoma/ przejrzystość zakodowanego programu,
- niemal nieuniknione, liczne omyłki w programie,
- konieczność bezproduktywnej pracy maszyny przy uruchamianiu programu.

Wobec tych trudności nic dziwnego, że problemy kodowania stanowiły dotąd jeden z najpoważniejszych czynników, hamujących szersze stosowanie elektronowych maszyn cyfrowych.

Stworzenie Systemu Automatycznego Kodowania SAKO, podobnie jak innych autokodów, stanowi próbę przewyciężenia tych trudności. W systemie takim programista zapisuje metodę rozwiązania problemu przez maszynę w specjalnym języku sformalizowanym, zbliżonym do zapisu stosowanego powszechnie w matematyce. Nauka i posługiwanie się tym językiem nie przedstawia specjalnych trudności. Na podstawie zapisu w SAKO maszyna sama koduje program w języku maszyny.

Programem zakodowanym w języku autokodu nazywamy jego zapis przy pomocy symboli i rozkazów uogólnionych, nie mających na ogół bezpośredniego znaczenia dla maszyny. Program zapisany w autokodzie musi być dopiero przetłumaczony na program w n i k o w y, zapisany w języku maszyny. Dokonuje tego specjalny program k o d u j ą c y, zapisany na stałe w pamięci bębnowej maszyny. W stosunku do programu kodującego program zapisany w autokodzie stanowi dane wejściowe, natomiast wynikiem jest ten sam program, zapisany w języku maszyny.

Program kodujący dla systemu SAKO składa się z około 5000 rozkazów w języku maszyny. Jego obszerność i złożoność spowodowana jest faktem, że zastępuje on całkowicie doświadczonego i sprawnego programistę.

Autokod pozwala zatem na szerokie stosowanie maszyn cyfrowych np. w przemyśle. Czas bowiem potrzebny na wyszkolenie personelu programującego maszynę w systemie SAKO jest wielokrotnie krótszy od czasu potrzebnego do wyszkolenia programistów, programujących w języku maszyny. Pozwala to specjalistom z różnych dziedzin

na przyswojenie sobie zasad pracy w autokodzie w ciągu kilku dni i tym samym otwiera szerokie perspektywy stosowania maszyn cyfrowych do analizy problemów powstających w trakcie prac badawczych.

W podręczniku zawarty jest opis języka SAKO, ułożonego dla maszyn XYZ i ZAM-2.

1. WSTĘPNE WIADOMOŚCI O MASZYNACH CYFROWYCH I PROGRAMOWANIU

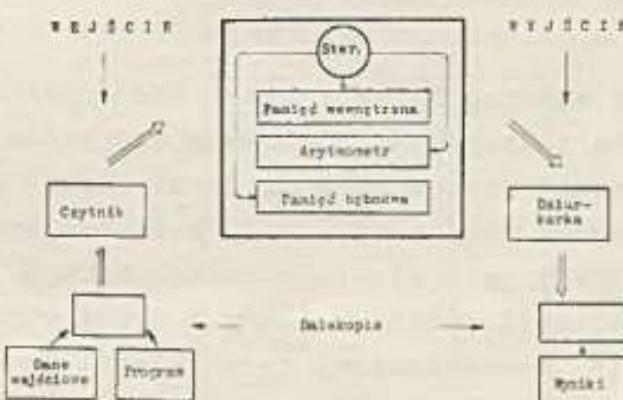
1.1. Schemat przetwarzania informacji

Automatyczne maszyny cyfrowe służą, w najogólniejszym ujęciu, do przetwarzania informacji wejściowych, d a n y c h - na wyjściowe, w y n i k i. Zarówno dane jak i wyniki zapisane są przy pomocy określonego typu znaków.

Przykładowo, w problemie rozwiązywania układu algebraicznych równań liniowych współczynniki tego układu oraz wyrazy wolne stanowią będą dane, zaś poszukiwane niewiadome - wyniki. Program powinien zawierać opis czynności potrzebny dla rozwiązania tego układu równań przez maszynę.

Obok problemów numerycznych, takich jak wyżej podany przykład, na maszynach cyfrowych możemy też rozwiązywać zadania innego typu. Jako informacja wejściowa może być np.

użyty pewien zwrot w języku polskim, jako informacja wyjściowa może być poszukiwany odpowiedni zwrot w języku francuskim. Program w tym przypadku powinien opisywać sposób tłumaczenia pewnych zwrotów przez maszynę z języka polskiego na francuski. Nie rozpatrując teraz zagadnienia praktycznych moż-



Rys. 1. Schemat przetwarzania informacji przez maszynę cyfrową

liwości ułożenia takiego programu w całej ogólności, można zaznaczyć, że przynajmniej teoretycznie, programy tego typu mogą być zapisane w języku SAKO.

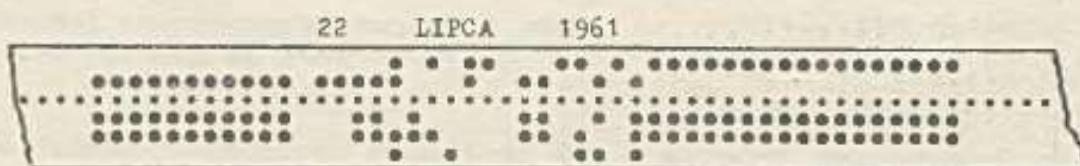
Obrazowy schemat przetwarzania informacji przez maszynę przedstawiony jest na rys. 1. Na wejściu maszyny został wyróżniony program i dane, na wyjściu zaś - wyniki. Niektóre szczegóły procesu przetwarzania danych na wyniki zostaną podane w następnym paragrafie.

1.2. Części składowe maszyn cyfrowych XYZ i ZAM-2

1.2.1. Urządzenia wejściowe

Zadaniem urządzeń wejściowych jest umożliwienie przejścia z języka cyfr, liter i innych znaków, przy pomocy których program i dane zapisane są przez programistę na formularzach, na język impulsów elektrycznych, jakim posługuje się maszyna.

W maszynach XYZ i ZAM-2 jako podstawowy środek pośredniczący została przyjęta pięciokanałowa taśma dziurkowana tego samego typu co taśma stosowana w międzynarodowej telegrafii dalekopisowej. W każdym rzędzie tej taśmy może być wydziurkowana pewna kombinacja dziurek; przy pięciu kanałach mamy $2^5 = 32$ różnych między sobą kombinacji. Odcinek taśmy z wydziurkowaną informacją "22 LIPCA 1961" przedstawiony jest na rys. 2.



Rys. 2. Odcinek taśmy z wydziurkowaną informacją "22 LIPCA 1961"

Dziurkowanie taśmy odbywa się przy pomocy dalekopisu. Jest to urządzenie zbliżone do zwykłej maszyny do pisania z tą różnicą, że równocześnie z naciśnięciem każdego klawisza następuje

obok zapisania litery na arkuszu wydziurkowanie na taśmie odpowiedniej kombinacji dziurek.

Zbiór znaków dalekopisowych znajdujących się na jego klawiaturze i ich odpowiedniość z kombinacjami dziurek na taśmie podany jest w tablicy 1. Jak wynika z tej tablicy, każdej kombinacji odpowiada

Tabela 1

ZNAKI DALEKOPISOWE DLA WEJŚCIA I WYJŚCIA NA FASMIE DZIURKOWANEJ

Cyfry	Litery	Taśma	Wartość binarna	Wartość dziesiętna
		5 4 3 2 1		
C y f r y				
1	A	.	0	0
2	B	• •	1	1
*	C	• • •	10	2
4	D	• •	11	3
(E	• • •	100	4
)	F	• • •	101	5
7	G	• • • •	110	6
8	H	• •	111	7
=	I	• • •	1000	8
-	J	• • •	1001	9
- .	K	• • • •	1010	10
v	L	• • •	1011	11
LINIA	M	• • -	1100	12
SP (spacja)	N	• • - •	1101	13
:	O	• • - • •	1110	14
0	P	• -	1111	15
>	Q	• - •	10000	16
:	R	• - •	10001	17
3	S	• - • •	10010	18
→	T	• - •	10011	19
5	U	• - • •	10100	20
6	V	• - • •	10101	21
/	W	• - • • •	10110	22
x	X	• - •	10111	23
9	Y	• - • -	11000	24
+	Z	• - • -	11001	25
L i t e r y				
.	.	• - • -	11010	26
?	?	• - • - •	11011	27
P.K.	£	• - • - •	11100	28
E/B/E/	E/B/E/	• - • - • •	11101	29
E/B/E/	E/B/E/	• - • - • •	11110	30
E/B/E/	E/B/E/	• - • - • •	11111	31

naogół jednocześnie jedna litera i jedna cyfra. Rozróżnienie między nimi określone jest znakiem "litera" lub "cyfra", zapisanym ostatnio na taśmie. Ilustracją tej reguły jest przykład, podany na rysunku 2.

Dziurkowanie taśmy przy pomocy dalekopisu posiada tę cenną zaletę, że wraz z taśmą otrzymujemy pisany dokument, będący dokładnym odpowiednikiem jej zawartości. Przy jego pomocy możemy np. wykryć niemal wszystkie błędy, powstałe przy dziurkowaniu taśmy. Na tej podstawie, posługując się otrzymaną taśmą i pewnymi dodatkowymi urządzeniami, możemy uzyskać nową, skorygowaną już taśmę. Dokument, odpowiadający skorygowanej taśmie, powinien stanowić wierną kopię części programu lub danych, zapisanych na formularzu wejściowym przez programistę.

Przykłady zapisów, uzyskanych przy pomocy dalekopisów znajdują się w dalszej części podręcznika w postaci programów, danych wejściowych i wyników.

Informacje, wydzielane na taśmie, przekazywane są maszynie za pośrednictwem czytnika, w którym następuje odpowiednie przesuwanie się taśmy pod źródłem światła. Znajdujące się na taśmie dziurki odsłaniają komórki fotoelektryczne, które, gdy padnie na nie światło, generują impulsy elektryczne.

Szybkość pracy czytnika jest znaczna i wynosi około 400 znaków na sekundę. Dzięki temu czas przeznaczony na wprowadzanie danych do maszyny jest stosunkowo krótki w stosunku do czasu potrzebnego na ich przygotowanie.

1.2.2. Urządzenia wejściowe

Urządzenia te stanowią odpowiedniki urządzeń wejściowych. Impulsy elektryczne pochodzące z maszyny i zawierające informacje o uzyskanych przez nią wynikach, powodują dziurkowanie taśmy na urządzeniu zwanym dziurkarką. Taśma ta z kolei może sterować dalekopisem, dzięki czemu możemy otrzymać wyniki w postaci pisemnej w formacie arkuszowym. Oczywiście, tym samym kombinacjom dziurek w rzadkach taśmy odpowiadają te same znaki daleko-

pisowe, co w urządzeniach wejściowych. W ten sposób np. jeden i ten sam dalekopis może być używany do dziurkowania taśmy i drukowania wyników. Ponadto, wyniki wydzielane na taśmie przez maszynę mogą być użyte jako dane wejściowe w następnych obliczeniach.

Szybkość pracy dziurkarki wynosi około 30 znaków na sekundę. Jest ona więc znacznie szybsza od dalekopisu, który drukuje z szybkością 7 znaków na sekundę. Z porównania tego wynika, że zastosowanie dziurkarki przed dalekopisem przyśpiesza wydawanie wyników przez maszynę.

1.2.3. Pamięć

Programy, dane wejściowe, wyniki pośrednie i końcowe przechowywane są w trakcie obliczeń w urządzeniu, zwany pamięcią maszyny.

Pamięć składa się z pewnej ilości podstawowych komórek, których zawartość informacyjną nazywamy słowem długim. Słowo takie może przedstawać sobą liczbę ułamkową lub słowo bułowskie /patrz następne rozdziały/. Każda połówka słowa długiego nosi nazwę słowa krótkiego; słowo krótkie może przedstawać rozkaz będący składnikiem programu lub liczbę całkowitą. Podział pamięci na słowa krótkie lub długie określony jest każdorazowo przez program.

Przeczytanie dowolnego słowa z komórki pamięci nie zmienia jej zawartości. Natomiast zapisanie słowa do komórki pamięci niszczy jej zawartość poprzednią.

Podstawowymi parametrami, charakteryzującymi urządzenia pamięciowe są:

- pojemność pamięci, to jest ilość zawartych przez nią podstawowych komórek,
- średni czas dostępu do pamięci, to jest czas średni, potrzebny na przeczytanie lub zapisanie słowa w przypadkowo dobranej komórce pamięci.

W maszynach XYZ oraz ZAM-2 dysponujemy dwoma rodzajami pamięci.

Pamięć wewnętrzna o pojemności 512 słów długich /1024 słów krótkich/ i średnim czasie dostępu około 0,0005 sekundy. Jest ona zbudowana na zasadzie rozchodzenia się fal ultradźwiękowych w furtach wypełnionych rtęcią.

Pamięć bębnowa o pojemności 8192 słów długich w XYZ i 16384 w ZAM-2 oraz średnim czasie dostępu do pojedynczego słowa około 0,02 sekundy. W przypadku czytania lub zapisywania bloku liczb, czas ten dotyczy pierwszego słowa bloku; czas dostępu do dalszych słów wynosi wówczas 0,002 sekundy. Zasadniczy element tej pamięci stanowi wirujący metalowy bęben, pokryty materiałem magnetycznym. Zapis informacji następuje tu przez odpowiednie namagnesowanie powierzchni bębna.

Wykonywanie wszystkich operacji przez maszynę, w tym operacji związanych z wejściem i wyjściem, odbywa się za pośrednictwem pamięci wewnętrznej. O ile cały program nie mieści się w tej pamięci, dzielimy go na mniejsze rozdziały, które w pamięci rtęciowej umieszczane są kolejno w odpowiednim porządku.

W pamięci bębnowej umieszczamy materiał liczbowy oraz te rozdziały programu, które nie są aktualnie opracowywane, a które są używane we wcześniejszych lub późniejszych etapach programu. Przesyłanie informacji z pamięci bębnowej do wewnętrznej lub odwrotnie kierowane jest przez program, znajdujący się w pamięci wewnętrznej. Wobec nadającego charakteru pamięci wewnętrznej w stosunku do bębnowej, ta ostatnia nosi często nazwę pamięci pomocniczej.

Najwygodniejszym dla programisty rozwiązaniem maszyny byłaby jedna szybka i dostatecznie pojemna pamięć wewnętrzna. Rozwiązanie takie byłoby jednak bardzo kosztowne. Dlatego też mało pojemną, lecz szybką pamięć wewnętrzną uzupełniamy w XYZ i ZAM-2 powolną, lecz za to bardzo pojemną pamięcią bębnową.

1.2.4. Arytmometr

Arytmometr jest urządzeniem, wykonującym w maszynie działania arytmetyczne lub logiczne na liczbach lub słowach bulowskich. Pracuje on bardzo szybko; czas dodania lub odjęcia dwóch liczb wynosi

około 0,0001 sekundy, czas zaś pomnożenia lub podzielenia - 0,003 sekundy. Do czasów tych dodaje się jednak czasy oczekiwania na pobranie rozkazów i liczb z pamięci wewnętrznej.

1.2.5. Sterowanie

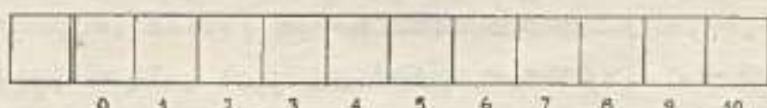
Sterowanie jest urządzeniem, koordynującym pracę pozostałych zespołów maszyny. Dodatkowo maszyna może być sterowana ręcznie za pośrednictwem przycisków i kluczy, znajdujących się na stoliku operatora.

1.3. Zapis dziesiętny liczb w maszynie

Jakkolwiek maszyny XYZ oraz ZAM-2 pracują w układzie binarnym, w języku SAKO przy zapisie liczb możemy się posługiwać systemem dziesiętnym.

1.3.1. Liczby ułamkowe

Dla zapisania liczby ułamkowej w pamięci maszyny potrzebna jest jedna komórka pamięci. Wyobrazić sobie możemy, że komórka ta składa się z 12 pozycji:



W pozycji oddzielonej dwoma kreskami jest zawsze wpisany znak liczby; w pozycjach następnych znajdują się kolejne cyfry dziesiętne danej liczby oraz kropka, oddzielająca część całkowitą od ułamkowej:

-	0	0	2	7	.	5	3	4	8	0	0
0	1	2	3	4	5	6	7	8	9	10	

Numer pozycji, w której jest wpisana kropka, nosi nazwę s k a l i d z i e s i e t n e j liczby, która sygnalizowana jest maszynie rozkazem USTAW SKALE DZIESIETNIE i deklaracją SKALA DZIESIETNA PARAMETROW.

Przykłady:

1. Liczba ułamkowa w skali 0:

+	*	7	8	9	5	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	

2. Liczba ułamkowa w skali 3:

-	0	1	2	*	3	4	5	6	0	0	0
0	1	2	3	4	5	6	7	8	9	10	

3. Liczba ułamkowa w skali 10:

-	0	0	0	3	1	3	2	8	4	5	*
0	1	2	3	4	5	6	7	8	9	10	

1.3.2. L i c z b y c a ł k o w i t e

Dla zapisania liczby całkowitej w pamięci maszyny potrzeba pół komórki pamięci. Wyobrazić sobie możemy, że połówka ta składa się z sześciu pozycji:

0	1	2	3	4	

W pozycji oddzielonej dwoma kreskami jest zawsze wpisany znak liczby całkowitej. W pozostałych pozycjach znajdują się cyfry dziesiątne liczby. Najmniej znacząca cyfra liczby znajduje się w ostatniej pozycji komórki. Np. liczba całkowita:

1281

jest zapisana w pamięci maszyny następująco:

+	0	1	2	8	1
0	1	2	3	4	

1.4. Zapis binarny liczb w maszynie

Podana wyżej interpretacja dziesiętna zawartości miejsc pamięci jest najwygodniejsza i zadawalająca dla większości obliczeń, przeprowadzanych w systemie dziesiętnym. Bliską rzeczywistej budowie maszyny jest jednak interpretacja binarna i posługiwanie się nią w wielu przypadkach może dać dodatkowe korzyści.

W interpretacji binarnej każdą komórkę pamięci można przedstawić jako ciąg 36 pozycji. W każdej z tych pozycji może być wpisana jedna z cyfr 0 i 1:

0	1	0	0	1	1	0	1	0	0	0	1	1	1	0	1	0	1	1	0	1	0	0	1	1	0	1	1	1	0	1	1	0			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Powstały w ten sposób ciąg 36 zer i jedynek nazywać będziemy dalej słowem binarnym lub krótko słowem. Zależnie od interpretacji, słowo możemy traktować jako liczbę ułamkową, liczbę całkowitą lub słowo bulowskie.

1.4.1. Liczby ułamkowe

Pozycja zerowa słowa określa znak liczby:

θ = liczba dodatnia

1 - liczba ujemna

Część całkowitą i ułamkową stanowią dwie bezpośrednio następujące po sobie części. Numer pozycji, w której znajduje się ostatnia cyfra części całkowitej nazywamy skalą binarną.

Diagram illustrating the structure of a floating-point number:

1	000011010	01100	.	0
---	-----------	-------	---	---

Annotations below the diagram:

- znak** (sign) points to the first bit.
- część całkowita** (integer part) points to the next 5 bits.
- część ułamkowa** (fractional part) points to the remaining 23 bits.

Powyższy przykład przedstawia liczbę ułamkową

-26.375

zapisaną w systemie binarnym w skali binarnej 9.

Jeśli część ułamkowa liczby rozpoczyna się od 1 pozycji, mówimy, że liczba jest zapisana w skali 0.

Skala binarna nie jest zaznaczona w maszynie przy zapisie liczb ułamkowych, lecz uwzględniana jest przez odpowiedni sposób wykonywania działań na tych liczbach. Skala binarna jest sygnalizowana maszynie rozkazem USTAW BINARNIE SKALE oraz deklaracją SKALA BINARNA PARAMETROW, np.

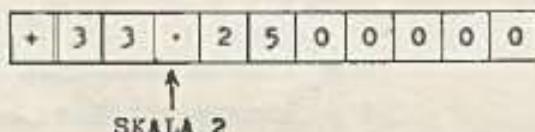
USTAW BINARNIE SKALE : 12

SKALA BINARNA PARAMETROW 8

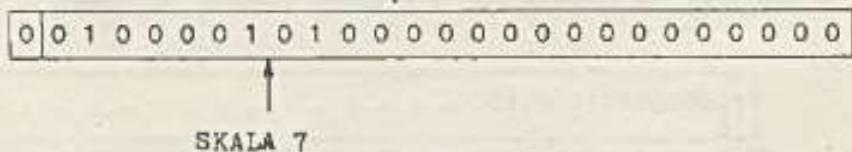
Każdej skali dziesiętnej odpowiada skala binarna według następującej tabelki:

SKALA DZIESIETNA	0	1	2	3	4	5	6	7	8	9	10
SKALA BINARNA	0	4	7	10	14	17	20	24	27	30	35

Tak więc liczba, zapisana w skali dziesiętnej 2:



w maszynie jest wyrażona następująco:



1.4.2. L i o z b y c a l k o w i t e

Na zapisanie liczby całkowitej w maszynie wykorzystujemy tylko połowę słowa, tzn. 18 pozycji dwójkowych. W ten sposób w jednej komórce pamięci możemy zapisać równocześnie dwie liczby całkowite.

Pierwsza z osiemnastu pozycji, w których zapisana jest liczba ułamkowa, służy do zapisania znaku liczby. Ostatnia cyfra znacząca w rozwinięciu dwójkowym danej liczby jest zapisana w ostatniej z rozpatrywanych 18 pozycji.

Przy wykonywaniu działań na liczbach całkowitych, maszyna traktuje je tak, jak gdyby zapisane one były w 36 pozycjach słowa, przy czym w pozycjach od 18 do 35 wpisane były zera. Na przykład liczbe 39 maszyna traktuje jako następujące słowo:

Utworzenie w maszynie liczby całkowitej, większej od

131 071 ($= 2^{17} - 1$)

powoduje powstanie nadmiaru i wszystkich jego konsekwencji /patrz rozkaz GDY NADMIAR/.

1-4-3. *Snowia* 1 *dziastanica* *buhowskiae*

Przez słowo bulowskie rozumiemy słowo, którego kolejne cyfry 0 lub 1 traktujemy jako elementy dwuelementowej algebry Boole'a. W szczególności, jako słowo bulowskie możemy traktować każda zapisana w maszynie liczba ułamkowa lub całkowite.

Zgodnie z określeniami działań w algebrze Boole'a wprowadzamy dla słów bulowskich pojęcie identyczności oraz działania sumy bulowskiej, iloczynu bulowskiego i negacji. Ponadto określamy działanie przesunięcia cyklicznego słowa. Sens tych działań objaśniony jest na przykładach, w których dla uproszczenia przyjęto, że słowo składa się tylko z pięciu cyfr.

a/ I d e n t y c z n o s ć

Zapis:

$$A \equiv B$$

Przykład:

$$\begin{array}{c} \boxed{0 \ 1 \ 0 \ 1 \ 1} \\ A \end{array} \quad = \quad \begin{array}{c} \boxed{0 \ 1 \ 0 \ 1 \ 1} \\ B \end{array}$$

Znaczenie:

Dwa słowa są identyczne, jeśli wszystkie cyfry jednego słowa są identyczne z odpowiadającymi im cyframi drugiego słowa.

b/ S u m a b u l o w s k a

Zapis:

$$A \equiv B + C$$

Przykład:

$$\begin{array}{c} \boxed{0 \ 1 \ 1 \ 1 \ 1} \\ A \end{array} \quad = \quad \begin{array}{c} \boxed{0 \ 1 \ 0 \ 1 \ 1} \\ B \end{array} \quad + \quad \begin{array}{c} \boxed{0 \ 1 \ 1 \ 0 \ 1} \\ C \end{array}$$

Znaczenie:

Wynikiem sumy bulowskiej dwóch słów jest słowo bulowskie, w którym na danej pozycji znajduje się cyfra 1 wtedy i tylko wtedy, gdy na tej pozycji znajduje się cyfra 1 w jednym lub w drugim składniku sumy.

c/ I l o c z y n bulowski

Zapis:

$$A \equiv B \times C$$

Przykład:

$$\begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 0 & 0 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 0 & 1 & 0 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 1 & 0 & 1 \\ \hline \end{array}$$

A B C

Znaczenie:

Wynikiem iloczynu bulowskiego dwu słów jest słowo bulowskie, w którym na danej pozycji znajduje się cyfra 1 wtedy i tylko wtedy, gdy na tej pozycji znajduje się cyfra 1 zarówno w jednym, jak i w drugim czynniku iloczynu.

d/ N e g a c j a

Zapis:

$$A \equiv -B$$

Przykład:

$$\begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 1 & 0 & 1 \\ \hline \end{array} = - \begin{array}{|c|c|c|c|c|} \hline 1 & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

A B

Znaczenie:

Wynikiem negacji słowa bulowskiego jest słowo bulowskie, w którym na danej pozycji znajduje się cyfra 1 wtedy i tylko wtedy, gdy na tej pozycji w słowie negowanym znajduje się cyfra 0.

e/ Przesunięcie cykliczne w lewo

Zapis:

$$A = B * n \quad n - \text{liczba całkowita dodatnia}$$

Przykład:

$$\begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 0 & 0 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 1 & 0 \\ \hline \end{array} * 2$$

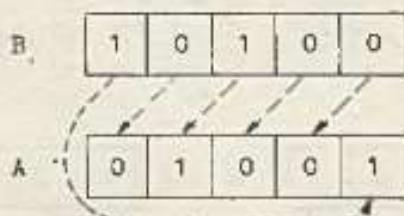
A B n

Znaczenie:

Wynikiem przesunięcia słowa B w lewo o jeden:

$$A \equiv B * 1$$

jest słowo binarzowe A, powstałe z przepisania słowa B tak, że każda cyfra stojąca w słowie B na pozycji m znajduje się w słowie A na pozycji m-1. Cyfra, stojąca w słowie B na zeroowej pozycji, zostaje przepisana w słowie A na ostatniej pozycji:



Przesunięcie cykliczne słowa o n w lewo, jest równoważne n-krotnemu przesunięciu o jeden w lewo.

f/ Przesunięcie cykliczne w prawo

Zapis:

$$A \equiv B * (-n) \quad n - \text{liczba całkowita dodatnia}$$

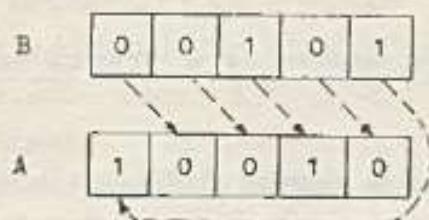
Przykład:

Znaczenie:

Wynikiem przesunięcia słowa B w prawo o jeden:

$$A \equiv B * (-1)$$

jest słowo bulowskie A, powstałe z przepisania słowa B tak, że każda cyfra, stojąca w słowie B na pozycji m znajduje się w słowie A na pozycji $m+1$. Cyfra, stojąca w B na ostatniej pozycji zostaje przepisana w słowie A na zerowej pozycji:



Przesunięcie cykliczne słowa o n w prawo jest równoważne n -krotnemu przesunięciu słowa o jeden w prawo.

ZAPIS OKTALOWY SŁÓW BULOWSKICH

Aby skrócić zapis zewnętrzny słowa bulowskiego, stosujemy tak zwany zapis oktalowy. Polega on na zastąpieniu kolejnych trójkę cyfr 0, 1 przez cyfry 0, 1, 2, 3, 4, 5, 6, 7 według następującej tabelki:

000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Tak więc słowo:

001|011|000|010|110|111|011|000|100|010|101|110

ma w kodzie oktalowym postać następującą:

130267304256

Aby nadać temu zapisowi większą przejrzystość, między dowolnymi kolejnymi grupami cyfr możemy stawiać kropki, np.

130.267.304.256

Nazwa "system oktalowy" pochodzi stąd, że dla zapisania przy użyciu jednego znaku dowolnej trójki zer i jedynek potrzeba ośmiu znaków.

1.5. Ogólna struktura programów w języku SAKO

Programy, napisane w języku SAKO, składają się z ciągu zdań. Rozróżniamy kilkadziesiąt różnych typów zdań SAKO, z których każdy ma określone znaczenie oraz ustaloną formę, a posługiwanie się każdym zdaniem ujęte jest w szereg regul. Język SAKO jest więc językiem sformalizowanym.

Przetwarzanie informacji przez maszynę składa się z dwóch zasadniczych etapów:

- wprowadzenie programu do maszyny,
- wykonanie wprowadzonego programu przez maszynę.

Z tego punktu widzenia dzielimy wszystkie zdania SAKO na następujące kategorie:

Deklaracje - podają informacje dotyczące budowy programu oraz ustalają znaczenie używanych dalej symboli. Deklaracje spełniają swą rolę przy wprowadzaniu programu do maszyny, są natomiast pomijane przy wykonywaniu programu przez maszynę.

Rozkazy - określają czynności maszyny w trakcie wykonywania programu.

Komentarze - mają jedynie charakter objaśnień poszczególnych fragmentów programu i są pomijane zarówno przy wprowadzaniu jak i przy wykonywaniu programu przez maszynę.

Należy rozróżnić kolejność wypisania rozkazów przez programistę od kolejności wykonania tych rozkazów przez maszynę. Naogół różnią się one między sobą.

Kolejność wypisania rozkazów, obok innych zdań SAKO zgodna jest z kolejnością ich wprowadzania do maszyny.

Kolejność wykonania rozkazów określona jest następująco:

- pierwszym rozkazem, wykonanym przez maszynę jest pierwszy rozkaz, wypisany w programie;
- po wykonaniu dowolnego rozkazu, który nie jest rozkazem sterującym, maszyna przechodzi do rozkazu następnego, to jest najbliższego w kolejności wypisania;
- przejście do rozkazu o innej kolejności wypisania nastąpić może tylko po rozkazach sterujących, przy spełnieniu określonych warunków i w sposób właściwy takiemu rozkazowi.

Ostatnim wypisanym zdaniem programu jest zawsze deklaracja KONIEC, sygnalizująca koniec wprowadzania programu do maszyny.

Ostatnim wykonanym rozkazem jest rozkaz STOP, sygnalizujący zakończenie wykonywania programu i zatrzymanie się maszyny.

Wszystkie programy SAKO dzielimy na rozdziały, stanowiące odcinki programu, które wraz z przyporządkowanymi im danymi mieszczą się jednocześnie w pamięci wewnętrznej maszyny.

Składnia zdań SAKO jest następująca.

Zdania SAKO mogą być zbudowane z następujących znaków, dostępnych na dalekopisie /tablica 1/:

1. Alfabet łaciński, składający się z 26 liter

A, B, C, D, E, F, G, ..., Z.

2. Cyfry

0 1 2 3 4 5 6 7 8 9

3. Znaki działań i relacji

+ - × / * = ≡ >

4. Seperatory i nawiasy

· , : ()

5. Spacje /odstępy między znakami/.

Pewne określone ciągi znaków /nie licząc spacji/ nazywać będziemy z w r o t a m i języka SAKO. Przykładami zwrotów są:

CZYTAJ:

NASTĘPNY

GDY

Pewne ciągi znaków o określonej b u d o w i e nazywać będziemy w y r a ż e n i a m i. Przykładami wyrażeń są:

liczby

zmienne

funkcje

listy zmiennych

wyrażenia arytmetyczne

wyrażenia bułowskie.

Pewne ciągi znaków o dowolnej budowie lecz jednoznacznie ograniczone, nazywać będziemy t e k s t a m i:

X1 = PIERWIASTEK

Zdania SAKO zbudowane są jako określone ciągi zwrotów, wyrażeń i tekstów. Rodzaj zwrotów, rodzaj wyrażeń, ograniczenia tekstów i kolejność ich występowania muszą być zgodne z jednym z dopuszczalnych typów zdań SAKO.

Każde wyrażenie SAKO rozpoczyna się od nowego wiersza. Ilość wierszy zajętych przez zdanie wynika wyłącznie z jego budowy. Naogół zdania SAKO zamykają się w jednym wierszu.

1.6. Metodyka przygotowania programu

Dla uzyskania pełnej przejrzystości programu zalecane jest, aby każdy program, przygotowany do pracy na maszynę, był zaopatrzony w podane poniżej pozycje:

1. pełne sformułowanie problemu,
2. metoda rozwiązań uwzględniająca specyfikę pracy maszyn cyfrowych,
3. dyskusja skal i występujących w problemie danych, wartości pośrednich i wyników,
4. dyskusja dokładności uzyskanych wyników,
5. postać danych, wprowadzanych do maszyny,
6. postać, w jakich chcemy uzyskać wyniki,
7. sieć działań,
8. objaśnienia sieci działań,
9. właściwy program w języku SAKO,
10. objaśnienia programu,
11. metoda sprawdzenia poprawności programu,
12. metoda sprawdzenia poprawności wyników,
13. sprawdzenie ilości miejsc pamięci zajętych przez program i jego poszczególne rozdziały.
14. obliczenie czasu wykonywania programu w różnych wariantach jego wykonania.

Po sprawdzeniu programu, jego wykonaniu na maszynie i sprawdzeniu wyników następuje jeszcze

15. ostateczne opracowanie dokumentacji rozwiązania problemu.

Jak widać z powyższego, napisanie właściwego programu stanowi tylko jedną z wielu pozycji, jakie powinien opracować programista.

2. FORMA I OPIS POJĘĆ PODSTAWOWYCH W JĘZYKU SAKO

Poniżej podany jest opis i forma tych wyrażeń, które będą wykorzystane w opisie rozkazów SAKO /rozdz. 3/ i które posiadają jednakowe znaczenie, niezależnie od rozkazu, w którym będą one używane.

2.1. Liczby

2.1.1. L i c z b a u ła m k o w a

Przykłady:

45.678
.007
-2345.
+2.7182818284
123455789.1

Znaczenie:

Ogólnie przyjęte w matematyce. Kropka, zwana kropką pozycyjną, rozdziela część całkowitą i ułamkową liczby.

Forma:

Ciąg, złożony z co najwyżej dziesięciu cyfr dziesiętnych oraz kropki, która może być umieszczona na początku, na końcu lub też pośrodku liczby.

Nie są liczbami ułamkowymi w sensie SAKO:

3.1415926536	jedenaście cyfr dziesiętnych
-34,56	znak , zamiast

Reguły:

1. Każdej liczbie ułamkowej w sensie SAKO odpowiada pewna skala, dziesiętna lub binarna, określająca sposób jej zapisania w komórce pamięci maszyny. Ogólnie biologicznie, zapis w skali dziesiętnej N składać się może z N cyfr

przed kropką pozycyjną oraz z 10-N cyfr po kropce. Scisłe biorąc, wartość bezwzględna liczby ułamkowej, zapisanej w skali dziesiętnej N nie może osiągać lub przekraczać liczby a_N ; tabelkę a_N w zależności od N podajemy poniżej:

N	a_N
0	1
1	16
2	128
3	1024
4	15364
5	131072
6	1'048576
7	16 777216
8	134 217728
9	1073 741824
10	34359 738368

Skalę dziesiętną dla liczb ułamkowych, występujących explicite w programie, podaje deklaracja SKALA DZIESIETNA PARAMETROW. W przypadku przekroczenia zakresu a_N , odpowiadającego zadeklarowanej skali, następuje błędne wprowadzenie liczby do pamięci maszyny.

2. Każdą liczbę całkowitą, nie więcej jak dziesięciocyfrową, można zapisać jako ułamkową w sensie SAKO. Musi być ona wpisana do pamięci maszyny w skali dziesiętnej, zgodnej z wyżej podaną tabelką.

3. Wyniki działań arytmetycznych, dokonywanych przez maszynę, są wpisywane do pamięci maszyny w określonej skali. Skala ta, dziesiętna lub binarna, ustalona jest przez rozkaz USTAW SKALE DZIESIETNIE /lub USTAW SKALE BINARNIE/. Obliczanie wartości wyrażeń arytmetycznych oraz wczytywanie liczb rozkazem CZYTAJ przed ustawieniem skali prowadzi z zasady do błędnych wyników.

4.. Gdy w wyniku działania otrzymamy liczbę ułamkową, o większej ilości cyfr znaczących przed kropką niż na to pozwala przyjęta skala, powstaje tzw. nadmiar połączony z błędny zapisem liczby w odpowiedniej komórce pamięci. Powstanie nadmiaru jest w

maszynie sygnalizowane przez automatyczne wpisanie liczby 1 do specjalnego rejestru, zwanego wskaźnikiem nadmiaru. Stan wskaźnika nadmiaru może być wykryty w maszynie przez rozkaz sterujący GDY NADMIAR. Wykonanie tego rozkazu powoduje m.in. wyzerowanie wskaźnika nadmiaru.

2.1.2. Liczba całkowita

Przykłady:

$$\begin{array}{r} 321 \\ -7 \\ \hline 45678 \\ +15 \\ \hline -7654 \end{array}$$

Znaczenie:

Ogólnie przyjęte w matematyce.

Forma:

Ciąg co najwyżej pięciu cyfr dziesiętnych. Liczby ujemne są poprzedzone znakiem - (minus). Liczby dodatnie mogą /ale nie muszą/ być poprzedzone znakiem + (plus).

Nie są liczbami całkowitymi w sensie SAKO:

987654	zapis sprzeczny z formą: 6 cyfr
345.	zapis sprzeczny z formą: obecność w ciągu kropki pozycyjnej.

Reguły:

1. Wynikiem działań arytmetycznych /z wyjątkiem dzielenia/ na liczbach całkowitych jest zawsze liczba całkowita.

2. Jeśli w wyniku pewnych działań powstanie w maszynie liczba całkowita nie mieściąca się w komórce pamięci /jest nią każda liczba, większa od 131 071/, powstaje nadmiar z wszystkimi jego konsekwencjami.

2.1.3. Słowo bulowskie

Przykłady:

123.456.701.234
777000174000
012345.670123
137.237
7773.41
566331

Znaczenie:

Przez słowo bulowskie rozumiemy ciąg 36 lub 18 cyfr 0 lub 1, które traktujemy jako elementy dwuelementowej algebra Boole'a. Dla zapisu tych słów przyjmujemy formę oktalową. Słowo bulowskie długie zawiera 36 zer lub jedynek, słowo bulowskie krótkie zawiera 18 zer lub jedynek.

Forma:

Słowo bulowskie długie:
Ciąg dwunastu cyfr od 0 do 7, przedzielonych kropkami.

Słowo bulowskie krótkie:
Ciąg sześciu cyfr od 0 do 7 przedzielonych kropkami. Kropka nie może stać na początku ani na końcu słowa.

2.1.4. Blok liczbowy

Znaczenie:

Przez blok rozumiemy skończoną i uporządkowaną grupę liczb, oznaczoną wspólnym symbolem. Liczby, wchodzące w skład bloku noszą nazwę elementów bloku. Każdemu elementowi bloku odpowiada układ indeksów, wyznaczających jego położenie w bloku. Ilość indeksów nazywa się wymiarem bloku, ilość kolejnych /począwszy od zera/ liczb naturalnych, przebieganych przez pewien indeks, nosi nazwę zakresu tego indeksu.

Przykłady:

1. Ponumerowana grupa 40 liczb:

$$a_0, a_1, a_2, \dots, a_{39}$$

stanowi przykład jednowymiarowego bloku zwanego wektorem. Każdy element wektora jest opatrzony jednym indeksem, którego zakres jest tutaj równy 40.

2. Układ liczb, tworzący tablicę o trzech wierszach i czterech kolumnach

$$\begin{array}{cccc} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \end{array}$$

stanowi dwuwymiarowy blok, zwany macierzą 3x4. Każdy element macierzy jest opatrzony dwoma indeksami, których zakresy w tym przypadku wynoszą 0..4.

W języku SAKO jest możliwe posługiwanie się blokami o dowolnej liczbie wymiarów. Każdy blok w programie musi być określony deklaracją BLOK, STRUKTURA, TABLICA lub TABLICA OKTALNA, które określają jego wymiar i zakresy indeksów.

Rozmieszczenie bloków w pamięci maszyny.

Każdy blok jest wprowadzany lub wczytywany do pamięci maszyny "wierszami" w kolejności, którą można określić, jak następuje: niech blok A będzie blokiem n-wymiarowym i

$$A_{i_1, i_2, \dots, i_n}$$

pewnym jego elementem. Niech ponadto zakresami indeksów

$$i_1, i_2, \dots, i_n$$

będą odpowiednio liczby

$$d_1, d_2, \dots, d_n.$$

Wówczas element ten będzie wprowadzony lub wczytany do pamięci ma-

szyny jako k-ty z kolej, licząc od zera, gdzie

$$k = \left(\dots \left((i_1 \cdot d_2 + i_2) \cdot d_3 + i_3 \right) \cdot d_4 + \dots + i_{n-1} \right) \cdot d_n + i_n.$$

Na przykład, w przypadku czterowymiarowego bloku o zakresach indeksów 3, 2, 4, 5, element np. wyznaczony przez indeksy

1, 1, 2, 4

będzie wczytany do pamięci maszyny jako

$$((1 + 2 + 1) \cdot 4 + 2) \cdot 5 + 4 = 74$$

z kolej, licząc od zera.

Znajomość kolejności wczytywania elementów bloków do pamięci maszyny jest konieczna przy przygotowywaniu danych wejściowych dla maszyny.

Ogólne uwagi o blokach

1. Blok może być utworzony albo wyłącznie z liczb całkowitych, albo wyłącznie z liczb ułamkowych, albo słów bulowskich krótkich, albo słów bulowskich długich.

2. W języku SAKO rozpatrujemy tylko bloki tzw. "prostokątne", to znaczy takie, że każdy indeks przebiega ustalony zbiór wartości niezależnie od wartości innych indeksów.

3. Nazwa bloku jest zbudowana zgodnie z zasadami budowy zmiennych prostych.

2.2. Zmienne

2.2.1. Z m i e n n a p r o s t a

Przykłady:

A
A12
EPSILON
R MALE
X3
SUMA KWADRATOW

Znaczenie:

Jest to symbol, przyjmujący w trakcie obliczeń różne wartości liczbowe.

Te same zmienne, użyte w różnych rozdziałach, posiadają całkowicie niezależne znaczenia. W tym samym rozdziale, zmienne, użyte w programie głównym i poszczególnych podprogramach są również od siebie niezależne.

Uwaga:

Każda zmienna prosta w danym programie ma przypisaną sobie komórkę pamięci maszyny, w której są wpisywane kolejne wartości tej zmiennej.

Forma:

Ciąg liter, cyfr i spacji, rozpoczynający się od litery. Dwie zmienne języka SAKO są traktowane jako identyczne, jeśli ich pierwsze cztery znaki, nie licząc spacji, są identyczne.

Np. zmienne:

WARIACJA
WARIANCJA

są w języku SAKO identyczne.

Rozróżnialne są natomiast zmienne:

R MALE
R MACIERZY ,

gdzie różnią się one czwartym z kolei znakiem: L i C.

Tak więc, własność identyfikującą posiadają cztery pierwsze znaki.

Nie są w SAKO zmiennymi:

A·B3	}	w ciągu występują nie tylko
C/8		litery i cyfry
8DF	}	ciąg rozpoczyna się od cyfry
3H8		

2.2.2. Zmienna indeksowana

Przykłady:

A(3)
BETA(0,8)
C8(2,BETA)
TLX(1,0,I,3,21)

Znaczenie:

Zmienna indeksowana jest to symbol elementu bloku, wyznaczony wartościami indeksów. Właściwości zmiennej indeksowanej nie różnią się od właściwości zwykłych prostych.

Forma:

Symbol zmiennej indeksowanej jest identyczny z nazwą bloku, którego element zmienna ta określa. Po symbolu tym występuje para nawiasów, wewnętrz której znajdują się indeksy rozdzielone przecinkami. Właściwość identyfikującą posiadającą cztery pierwsze znaki symbolu zmiennej /nie licząc spacji/.

Reguły:

1. Posługiwianie się zmiennymi indeksowanymi w programie wymaga uprzedniej deklaracji BLOK lub STRUKTURA, odnoszącej się do właściwych bloków.
2. Indeksy przyjmować mogą jedynie wartości całkowite. Indeksami zmiennej indeksowanej mogą więc być tylko
 - liczby całkowite nieujemne,
 - zmienne proste lub indeksowane o wartościach całkowitych nieujemnych,
 - wyrażenia arytmetyczne o wartościach całkowitych nieujemnych /patrz wyrażenie arytmetyczne str. 36/.

2.3. Numery

Przykłady:

1
1 WYJSCIE
2BC .
789
3XG5

Znaczenie:

Jest to symbol, służący do oznaczania rozkazów SAKO. Numer wypisuje się z lewej strony rozkazu, oddzielając go od rozkazu nawiasem zamykającym. O rozkazie, przy którym został napisany numer α , mówimy, że jest on opatrzony numerem α , lub że jest to rozkaz o numerze α . Nie jest konieczne, aby wszystkie rozkazy programu były opatrzone numerami. Dwa rozkazy mogą być opatrzone tymi samymi numerami jedynie wówczas, gdy rozkazy te nie występują wewnątrz tego samego paragrafu lub rozdziału.

Forma:

Ciąg liter, cyfr i spacji, rozpoczynający się od cyfry. Własność identyfikującą posiadają cztery pierwsze znaki, nie licząc spacji.

2.4. Funkcje

Przykłady:

SIN (X + A)
TRY (ALFA)
TNG (X)

Znaczenie:

Znaczenie ogólnie przyjęte w matematyce. W języku SAKO dopuszczalne są dwa typy funkcji

- a. Funkcje języka
- b. Funkcje definiowane /patrz str. 81/

W SAKO wyróżniamy następujące funkcje języka:

Tablica 2

Skrot	Nazwa	Znaczenie
SIN(X)	Sinus	
COS(X)	Cosinus	
TNG(X)	Tangens	
ASH(X)	Arcus sinus	
ACS(X)	Arcus cosinus	
ATG(X)	Arcus tangens	
ARC(X,Y)	Arcus	Oznacza kąt, leżący w ćwiartce kątowej wyznaczonej przez znak X i znak Y, którego tangens jest równy Y/X.
PWK(X)	Pierwiastek kwadratowy	
PWS(X)	Pierwiastek sześcienny	
LN(X)	Logarytm naturalny	
EXP(X)	Eksponent	
MAX(X,Y,...,Z)	Maksimum	Oznacza największą z liczb X,Y,..,Z
MIN(X,Y,...,Z)	Minimum	Oznacza najmniejszą z liczb X,Y,..,Z
AOD(X,Y)	Moduł	Oznacza resztę powstałą z dzielenia X przez Y /tylko dla całkowitych A i B/
SGN(X,Y)	Signum	Oznacza liczbę $ X \cdot \operatorname{sgn} Y$
ABS(X)	Wartość bezwzględna /absolutna/	Oznacza wartość absolutną X
CNT(X)	Część całkowita	Oznacza część całkowitą liczby X

Inne funkcje jako funkcje definiowane wprowadza się rozkazem PODPROGRAM.

Forma:

Funkcje języka zapisuje się symbolem podanym obok funkcji. Argumenty podaje się w nawiasach, oddzielając je przecinkami.

Nazwy funkcji definiowanych są tworzone zgodnie z zasadami przyjętymi dla zmiennych prostych z tym, że własność identyfikującą posiadają tutaj trzy pierwsze litery nie licząc spacji. Argumenty funkcji definiowanych podaje się w sposób identyczny, jak dla funkcji języka SAKO.

Uwagi:

- Trzy pierwsze litery nazwy funkcji definiowanej nie mogą pokrywać się z trzema pierwszymi literami funkcji języka.
- Sześć ostatnich funkcji, podanych w tablicy 2 można używać jedynie w wyrażeniach arytmetycznych; na przykład:

$A = CAŁKA(A, B, ABS(), EPSILON)$ źle
 $A = CALKA(A, B+ABS(c), SIN(), 0.01)$ dobrze

2.5. Wyrażenia arytmetycznePrzykłady:

$A + 1$
 $A + ALFA/X15 + . 7321$
 $GAMMA - D3(A + B, 9.81)$
 $LOG(SIN(X + Y) / PWK(Z))$
 $A(3, B(4, J), ENT(C + D)) + S$
 $SIGMA/B + SIGMA * .33 + 1$

Znaczenie:

Wyrażenie arytmetyczne posiada sens ogólnie przyjęty w matematyce. Znaki $+$ $-$ \times $/$ posiadają konwencjonalne znaczenia dodawania, odejmowania, mnożenia i dzielenia. Znak $*$ jest symbolem potęgowania:

$$A * B \text{ oznacza } A^B.$$

Nawiąsy określają – jak zazwyczaj – kolejność wykonywania działań, zamykają argumenty funkcji lub też zamykają indeksy zmiennych indeksowanych.

Każde wyrażenie arytmetyczne ma jednoznacznie określoną wartość, powstającą w wyniku wykonywania działań, wymienionych w wyrażeniu. Wartością wyrażenia może być liczba całkowita lub ułamkowa.

Forma:

Sposób budowy wyrażenia arytmetycznego określamy rekurencyjnie w sposób następujący:

- Zmienna bądź liczba jest wyrażeniem.
- Jeśli A jest wyrażeniem, wówczas (A) jest również wyrażeniem.

3. Jeśli A jest wyrażeniem nie rozpoczynającym się od znaku + lub -, wówczas -A jest również wyrażeniem.
 4. Jeśli A, B są wyrażeniami, B nie rozpoczyna się od znaku + lub -, wówczas

$$\begin{aligned} & A + B \\ & A - B \\ & A \times B \\ & A / B \\ & A * B \end{aligned}$$

są również wyrażeniami.

5. Jeśli G jest nazwą bloku, A, B, ..., Z są wyrażeniami przyjmującymi wartości całkowite i ilość tych wyrażeń jest równa ilości indeksów danej zmiennej indeksowanej G, wówczas

$$G(A, B, \dots, Z)$$

jest również wyrażeniem.

6. Jeśli f jest nazwą funkcji, A, B, ..., Z są wyrażeniami w ilości i rodzaju zgodnym z argumentami tej funkcji, wówczas

$$f(A, B, \dots, Z)$$

jest również wyrażeniem.

Powyzsza definicja formy wyrażenia pozwala budować wyrażenia drogą składania ich z wyrażeń mniej skomplikowanych, bądź też pozwala sprawdzać poprawność budowy konkretnych wyrażeń arytmetycznych.

Nie są wyrażeniami:

$$\begin{aligned} & A + / B \\ & A + -4.5678 \\ & A * * C \\ & (A + B) C \end{aligned}$$

Kolejność wykonywania działań w wyrażeniu arytmetycznym:

W SAKO przyjmujemy następującą listę moczy działań:

1. Obliczanie wartości funkcji i wartości zmiennych indeksowanych.
2. Wykonywanie działań * /potęgowanie/.
3. Wykonywanie działań × .

4. Wykonywanie działań /.
5. Wykonywanie działań -.
6. Wykonywanie działań +.

Działaniem o większej mocy nazywamy będącym działanie, stojące na podanej liście pod mniejszym numerem.

Mając określoną moc działań, możemy teraz podać ogólne reguły kolejności ich wykonywania.

Z dwu działań wykonuje się najpierw to, które jest zamknięte w większej ilości nawiasów.

W przypadku równej ilości nawiasów wykonuje się najpierw działanie o większej mocy.

Działania o równej mocy zamknięte w jednakowej ilości nawiasów wykonuje się kolejno od lewej do prawej strony wyrażenia arytmetycznego.

Wyrażenia całkowite i ułamkowe

Każde wyrażenie arytmetyczne SAKO posiada wartość całkowitą lub ułamkową, innymi słowy - jest całkowite lub ułamkowe. Decydują o tym następujące reguły, które stosuje się przy obliczaniu wartości wyrażenia:

1. Zmienne proste, indeksowane oraz funkcje mają wartości całkowite /lub krócej: są całkowite/ wtedy i tylko wtedy, gdy są wymienione w deklaracji CAŁKOWITE.
2. Funkcje ENT i SGN(A, B), gdy A jest całkowite, są z definicji funkcjami o wartościach całkowitych.
3. Jeśli A jest całkowite, to (A) i -A są całkowite. Analogiczna reguła dotyczy wyrażeń ułamkowych.
4. Wyrażenie A/B jest zawsze ułamkowe. Natomiast $A + B$, $A - B$, $A \times B$, $A * B$ są tylko wtedy całkowite, gdy zarówno A i B są całkowite.

Wynika stąd, że wyrażenie ma wartość całkowitą wtedy i tylko wtedy, gdy wszystkie występujące w nim

- liczby ułamkowe
- zmienne ułamkowe
- znaki dzielenia
- funkcje ułamkowe

znajdują się w wyrażeniu pod znakami funkcji o wartościach całkowitych.

Skala obliczania wartości wyrażenia

Jeśli wartość wyrażenia jest liczbą ułamkową, to jest ona obliczana zgodnie z ostatnio wykonanym rozkazem USTAW SKALE, ustalającym skalę obliczanych wartości.

Wszystkie stałe lub zmienne ułamkowe, występujące w wyrażeniu, muszą być wprowadzone, wczytane lub obliczone w tej samej skali, zgodnej z ostatnio wykonanym rozkazem USTAW SKALE DZIESIETNIE (BINARNIE) bądź ostatnio napisaną deklaracją SKALA DZIESIETNA (BINARNA) PARAMETROW.

Uwagi:

1. Wszystkie zmienne, występujące w wyrażeniu muszą mieć uprzednio określone wartości przez wprowadzenie, wczytanie lub obliczenie.
2. Para dodatkowych nawiasów, nie koniecznych dla jednoznacznego odczytania wyrażenia, nie stanowi błędного zapisu.
3. Znak \times dla mnożenia jest istotny. Nie można go zastępować znakiem \cdot lub pomijać.
4. Wszystkie wartości pośrednie, jak i końcowy wynik obliczeń, określonych w wyrażeniu, muszą mieścić się w zakresie aktualnie przyjętej skali. Przekroczenie zakresu skali w trakcie obliczania wartości wyrażenia arytmetycznego powoduje wpisanie liczby 1 do rejestru WSKAZNIK NADMIARU i może być wykryte w programie za pośrednictwem rozkazu sterującego GDY BYŁ NADMIAR.

2.6. Wyrażenia bulowskiePrzykłady:

$$\begin{aligned} & A + \text{BETA} \\ & A + (-C) \times 528 \\ & C * (-2) + \text{KSI} \\ & G \times 000.077.777.760 \\ & A \times (-B) + (-A) \times B \end{aligned}$$

Znaczenie:

Wyrażeniem bulowskim jest ciąg znaków, określający rodzaj i kolejność wykonywania działań bulowskich na słowach, których symbole występują w wyrażeniu. Każde wyrażenie bulowskie ma jednoznacznie określoną wartość, będącą również słowem bulowskim.

Forma:

Poniżej podajemy rekurencyjne określenie formy wyrażenia bulowskiego, które pozwala tworzyć i sprawdzać poprawność budowy wyrażenia:

1. Każda zmienna prosta lub słowo bulowskie jest wyrażeniem bulowskim.

2. Jeżeli A jest wyrażeniem bulowskim, nie rozpoczynającym się od znaku - , wówczas

$$-A$$

jest również wyrażeniem bulowskim.

3. Jeżeli A jest wyrażeniem bulowskim, wówczas

$$(A)$$

jest również wyrażeniem bulowskim.

4. Jeżeli A i B są wyrażeniami bulowskimi, zaś B nie rozpoczyna się od znaku - , wówczas

$$A + B$$

$$A \times B$$

są wyrażeniami bulowskimi.

5. Jeżeli A jest wyrażeniem, I - zmienna całkowita lub liczba całkowitą nieujemną, wówczas

$$A * I$$

$$A * (-I)$$

są wyrażeniami bulowskimi.

Reguły:

1. Wartość wyrażenia bulowskiego oblicza się według następującej hierarchii działań:

1. wykonywanie przesunięć *
2. wykonywanie iloczynów bulowskich × ,
3. wykonywanie sum i negacji + -

Z dwu działań wykonuje się najpierw to, które jest zamknięte w większej ilości nawiasów. W przypadku równej ilości nawiasów wykonuje się to, które jest wymienione w liście hierarchii we wcześniejszej kolejności. Działania, zamknięte w jednakową ilość nawiasów i stojące na liście hierarchii w tym samym punkcie, wykonuje się kolejno od lewej do prawej strony wyrażenia bulowskiego.

2. Każda liczba może być traktowana jako słowo bulowskie i odwrotnie. Wartością wyrażenia bulowskiego jest słowo bulowskie, które może być traktowane w dalszym ciągu programu jako liczba całkowita lub ułamkowa.

2.7. Lista

Przykłady:

A, B, C, D, E
 1, 2, 3, 4, 5, 6, 7
 AX4, BT, *ALFA, *GAMMA,
 1WYJSCIE, 2CA, 81CTB, -
 3BX, 8
 A(K+1), A(K+2), A(K+3), A(K+4)
 A, B, F(), GXB(), *BVN, ASD

Znaczenie:

Lista jest to ciąg zmiennych, liczb, numerów, nazw bloków lub podprogramów, oddzielonych przecinkami. Listę stosuje się w rozkazach lub deklaracjach, które mogą odnosić się do więcej niż jednej zmiennej, liczby bloku lub podprogramu.

Forma:

Symboli, występujące na liście, pisane są zgodnie z zasadami, dotyczącymi ich formy. Nazwy bloków, występujących na liście, po przedziale są symbolem * ; np.

* GAMMA

Nazwy podprogramów, występujących na liście, uzupełnione są parą nawiasów:

F()

Zasada ta nie obowiązuje w tych rozkazach lub deklaracjach, które odnoszą się z reguły do bloków lub podprogramów, tzn. gdy z góry wiadomo, jakiego charakteru symbole występują na liście. Np. w deklaracji STRUKTURA, na liście której występują z reguły *

tylko nazwy bloków, nie uzupełnia się tych nazw gwiazdkami:

STRUKTURA: (N, M, 6) : A, B, C

Listę można przenosić /i to wielokrotnie/ z jednego wiersza do drugiego. Stosuje się w tym celu znak - , który stawia się po przecinku, oddzielającym ostatni symbol w górnym wierszu od pierwszego symbolu w wierszu poniżej, np.:

WE, RTZ, UI, OP, ASDF, -
GHJK, KL, YXC

3. FORMA I OPIS POSZCZEGÓLNYCH ROZKAZÓW SAKO

3.1. Deklaracje

ROZDZIAŁ : nazwa

Przykłady:

ROZDZIAŁ : GAUSS - SEIDEL
ROZDZIAŁ : LICZENIE PIERWIĄSTKOW
ROZDZIAŁ : L2A

Znaczenie:

Część programu, jaką możemy w całości pomieścić w pamięci wewnętrznej maszyny, nazywamy rozdziałem.

Deklaracja ROZDZIAŁ określa wypisaną pod nią część programu aż do najbliższej deklaracji ROZDZIAŁ lub KONIEC jako rozdział i jednocześnie nadaje mu nazwę.

Forma:

nazwa - składa się z liter, cyfr i spacji, przy czym pierwszym znakiem jest litera. Własność identyfikująca posiada 10 pierwszych znaków /nie licząc spacji/.

Reguły:

1. Przed pierwszym wypisanym rozdziałem nie potrzeba dawać deklaracji ROZDZIAŁ, o ile nie jest potrzebny powrót do tego rozdziału /rozkaz sterujący IDZ DO ROZDZIAŁU/. W szczególności deklaracja ROZDZIAŁ jest zbędna w programach, mieszczących się w jednym rozdziale.

2. Wszystkie deklaracje tracą swoje znaczenie przy przejściu z jednego rozdziału do drugiego. Wartości zmiennych w jednym rozdziale przestają być aktualne w obrębie drugiego rozdziału. To samo dotyczy numerów rozkazów oraz nazw podprogramów. Tak więc zmiennym, użytym w jednym rozdziale muszą być nadane wartości w tym samym rozdziale. Podobnie podprogramy, używane w pewnym rozdziale muszą być w nim określone. Ostatni wypisany rozdział musi być zakończony deklaracją KONIEC, która sygnalizuje koniec wprowadzania programu.

3. Wykonywanie programu rozpoczyna się od pierwszego wypisanego rozdziału. Kolejność przechodzenia do dalszych rozdziałów określona jest rozkazami IDZ DO ROZDZIAŁU, a więc nie musi następować w kolejności ich wypisania.

Uwaga:

Wszystkie rozdziały, przełożone na język maszyny, zapisane są w pamięci bębnowej. Przy przejściu do nowego rozdziału rozdział ten zostaje przepisany z pamięci bębnowej do wewnętrznej na miejsce poprzednio wykonywanego rozdziału. Kosztuje to pewną ilość czasu, dlatego też należy w miarę możliwości unikać przechodzenia z jednego rozdziału do drugiego.

PODPROGRAM : lista wyników = nazwa (lista argumentów)

Przykłady:

PODPROGRAM: (U, V, W) = TRY(X, Y, Z)

PODPROGRAM: (*A) = MN MACIERZY (*B, *C, N, M, L)

PODPROGRAM: CALKA (A, B, F(), H)

Znaczenie:

Deklaracja ta określa ciąg następujących po niej rozkazów aż do najbliższej deklaracji PODPROGRAM, ROZDZIAŁ lub KONIEC jako podprogram o danej nazwie, argumentach i wynikach.

Forma:

Nazwa podprogramu jest budowana zgodnie z zasadami tworzenia zmiennych prostych, z tym, że własność identyfikującą posiadają tutaj trzy pierwsze znaki, nie licząc spacji. Lista wyników jest zamknięta w nawiasy i stoi po lewej stronie znaku = . Lista argumentów, również w nawiasach, znajduje się po prawej stronie znaku = .

Jako argumenty występuować mogą symbole:

liczb,
bloków,
podprogramów,
funkcji.

Jako wyniki występuować mogą symbole:

'liczb,
bloków.

Aby odróżnić nazwy bloków od zmiennych, na liście przed nazwami bloków stawiamy gwiazdkę. Aby odróżnić zmienne od nazw podprogramów i funkcji, po ich nazwach stawiamy parę nawiasów, np.

TRY()

3. W przypadku, gdy podprogram służy do określenia funkcji F, w deklaracji PODPROGRAM nie występuje lista wyników ani znak = /patrz ostatni z podanych przykładów/. W tym przypadku ostatni wykonywany rozkaz arytmetyczny podprogramu musi mieć postać

$F() = \dots$

gdzie prawa strona formuły powoduje obliczenie ostatecznej wartości funkcji.

Lewa strona formuły dla przykładu trzeciego ma postać

CALKA() = ...

Reguła:

Kolejność występowania na listach argumentów i wyników jest ustalona i nie może ulegać zmianom.

Uwagi:

1. Miejsce w pamięci maszyny na bloki, będące argumentami lub wynikami podprogramu, rezerwuje program główny. W podprogramie podaje się tylko strukturę tych bloków.
2. Funkcjami, które są argumentami podprogramu, mogą być zarówno funkcje języka /z wyjątkiem MAX, MIN, MOD, ABS, SGN, ENT/ jak i funkcje określone przez podprogramy.

BLOK (n, m, ..., k) : lista

Przykłady:

```
BLOK (15) : A, B
BLOK (3, B) : ALFA, G2
BLOK (2, 2, 2, 5, 3) : CX2
```

Znaczenie:

Deklaracja ta stwierdza, że symbole, wymienione w liście są nazwami bloków o wspólnej strukturze, określonej przez parametry deklaracji.

Format:

n; m, ..., k	- liczby naturalne
lista	- jest utworzona z nazw bloków.

Uwagi:

1. Deklaracja BLOK powoduje zarezerwowanie w pamięci wewnętrznej maszyny odpowiedniej ilości miejsc na pomieszczenie elementów bloków, wymienionych w liście.
2. Użycie każdej zmiennej indeksowanej i każdej deklaracji STRUKTURA musi być poprzedzone w programie właściwą deklaracją BLOK.

3. Jeżeli dwa różne rozdziały rozpoczynają się od deklaracji BLOK, określających analogiczne ciągi bloków o tej samej strukturze /choć nawet o różnych nazwach/, to wartości elementów tych bloków określone w jednym rozdziale, zachowują się przy przejściu do drugiego rozdziału.

Reguła ta staje się zrozumiałą, gdy uwzględnimy, że deklaracje BLOK rezerwują miejsca pamięci na dane zawsze w ten sam sposób i zawartość tych miejsc nie ulega zmianie przy przejściu z jednego rozdziału do drugiego.

Wyjaśnia to następujący przykład:

ROZDZIAŁ : ALFA

BLOK (99) : B, C

CZYTAJ: * B

C(J) = -----

Nadawanie wartości zmiennym indeksowanym określającym elementy bloków B i C

ROZDZIAŁ : BETA

BLOK(99) : D, E

W rozdziale ALFA deklaracja BLOK określa strukturę bloków B, C oraz rezerwuje $100 + 100 = 200$ miejsc w pamięci maszyny na pomieszczenie liczb, wchodzących w skład tych bloków. Wykonanie rozkazów: CZYTAJ : B oraz formuł arytmetycznych

$C(J) = \dots$

dla $J = 0, 1, \dots, 99$ powoduje wypełnienie zarezerwowanych miejsc liczbami /czyli powoduje nadanie zmiennym $B(I)$, $C(I)$ określonych wartości/.

W rozdziale BETA deklaracja BLOK powoduje również zarezerwowanie 200 miejsc w pamięci, określając jednocześnie strukturę bloków D i E. Jeżeli teraz, po wykonaniu rozdziału ALFA rozkazem IDZ DO ROZDZIAŁU : BETA przejdziemy do tego rozdziału, to końcowe wartości bloków B i C rozdziału ALFA staną się początkowymi wartościami bloków D i E rozdziału BETA.

STRUKTURA (I, J, ..., K) : lista

Przykłady:

STRUKTURA (ALFA) : A, B

STRUKTURA (N, S) : BETA: GJ

STRUKTURA (K, L, M) : CX2, B, D

Znaczenie:

Deklaracja ta zmienia strukturę bloków, wymienionych w liście, a zadeklarowanych napisaną we wcześniejszej kolejności deklaracją BLOK. Nowa struktura bloków jest utworzona przez podanie liczb lub zmiennych, określających wymiar bloków i zakresy ich indeksów.

Forma:

I, J, ..., K - zmienne lub liczby całkowite, określające wspólne zakresy indeksów wymienionych na liście bloków.

lista - jest utworzona z nazw bloków.

Uwagi:

1. Rezerwowanie miejsc w pamięci na pomieszczenie elementów bloków odbywa się jedynie za pośrednictwem deklaracji BLOK; deklaracja STRUKTURA nie posiada tej właściwości.
2. Ilość elementów bloku w zmienionej strukturze może być co najwyżej równa ilości miejsc pamięci zarezerwowanych przez właściwą deklarację BLOK.

TABLICA (n, m, ..., k) : nazwa

Przykłady:

TABLICA (2, 3) : A

3.7182	45.13	.1508
-.35	9.83	32.01
*		

TABLICA (7) : DZETA

21	4	694	37
0	293	23	
*			

Znaczenie:

Deklaracja ta określa nazwę i strukturę bloku, którego elementy są wypisane począwszy od następnego wiersza formularza. Blok ten wprowadzany jest do maszyny równocześnie z programem i znajduje się w tej części pamięci, w której mieści się program obliczeń.

Forma:

Struktura wymienionego bloku określona jest przez podanie po słowie TABLICA zakresów indeksów w ilości i kolejności odpowiadającej indeksowaniu elementów tego bloku. Zakresy te są zamknięte w nawiasy i przedzielone przecinkami.

Poczynając od następnego po deklaracji TABLICA wiersza formularza wypisuje się listę liczb według zasad przygotowywania danych wejściowych /patrz rozkaz CZYTAJ/.

Zakresy indeksów:

n, m, ..., k

są w deklaracji TABLICA podane w postaci nieujemnych liczb całkowitych.

N a z w a w deklaracji TABLICA jest napisana zgodnie z przyjętymi zasadami pisania nazw bloków.

Uwagi:

1. Deklaracja TABLICA wraz z następującym po niej układem liczb powoduje nadanie liczbowych wartości elementom wymienionego bloku w trakcie w p r o w a d z a n i a p r o g r a m u . Jest to istotna różnica w stosunku do deklaracji BLOK i STRUKTURA. Blokom, określonym tymi deklaracjami, wartości nadawane są dopiero w trakcie w y k o n y w a n i a p r o g r a m u .
2. Gdy tablica składa się z liczb ułamkowych, skala, w jakiej mają być wprowadzone jej elementy, musi być ustalona za pośrednictwem poprzednio wypisanej deklaracji, dotyczącej skali parametrów.
3. W trakcie wykonywania programu można zmieniać pierwotne wartości elementów tablicy.

TABLICA OKTALNA (n, m, ..., k) : nazwa

Przykłady:

TABLICA OKTALNA(5) : ALFA

074.562.321.777
 7777.0000.7777
 252525.252525
 400.000.000.000
 333.333.333.333
 000.760.003770

*

TABLICA OKTALNA(1, 2) : KAPPA

123456.710111
 34.72.27.61.23.41
 121126.123365
 7321.0012.375
 111.000.111.000
 721641002831

*

Znaczenie:

Deklaracja ta jest identyczna z deklaracją TABLICA z tym, że tablica oktalna utworzona jest ze słów bulowskich, zapisanych oktalnie.

Forma:

Analogiczna, jak w przypadku TABLICY z tym, że zamiast liczb występują słowa bulowskie.

Uwaga:

Patrz 1 i 2 uwaga dla deklaracji TABLICA.

CALKOWITE : lista

Przykłady:

CALKOWITE: A, GAMMA, X3

CALKOWITE: L, L1, * K2, TRY()

Znaczenie:

Deklaracja ta stwierdza, że wartości zmiennych prostych, elementów bloków lub wyników podprogramów, których nazwy występują na liście, są liczbami całkowitymi.

Forma:

Lista deklaracji CALKOWITE jest listą zmiennych, nazw bloków i nazw podprogramów. Słowo CALKOWITE oddzielone jest od pierwszego symbolu na liście dwukropkiem.

Reguły:

1. Zmienne, elementy bloków, wyniki podprogramów, których nazwy nie są wymienione w deklaracji CALKOWITE, przyjmują wartości ułamkowe. Tak więc deklaracja CALKOWITE dokonuje podziału zmiennych na całkowite i ułamkowe.

2. Deklaracja CALKOWITE traci znaczenie przy przejściu z jednego rozdziału do drugiego, a w obrębie jednego rozdziału, przy przejściu z programu głównego do podprogramu, lub z jednego podprogramu do drugiego.

SKALA DZIESIETNA PARAMETROW n

Przykłady:

SKALA DZIESIETNA PARAMETROW 0

SKALA DZIESIETNA PARAMETROW 3

Znaczenie:

Deklaracja ta ustala skalę dziesiętną, w jakiej będą w programie /wraz z programem/ wszystkie liczby ułamkowe /tak zwane parametry liczbowe/ w rozkazach o dalszej kolejności wypisania - aż do nowej deklaracji, dotyczącej skali parametrów.

Forma:

n - liczba naturalna, zawarta w przedziale $[0, 10]$.

Reguła:

W każdym programie pojawienie się liczb ułamkowych musi być poprzedzone deklaracją dotyczącą skali parametrów.

Uwaga:

Parametry liczbowe w programie występują jedynie

- w wyrażeniach arytmetycznych,
- w tablicach,
- w formułach arytmetycznych i operacyjnych.

SKALA BINARNA PARAMETROW n

Przykłady:

SKALA BINARNA PARAMETROW 0

SKALA BINARNA PARAMETROW 33

Znaczenie:

Deklaracja ta ustala skalę binarną liczb ułamkowych, wprowadzanych w dalszej kolejności - aż do nowej deklaracji, dotyczącej skali parametrów.

Forma:

n - liczba naturalna zawarta w przedziale $[0, 35]$.

Reguła:

Patrz - Reguła dla deklaracji SKALA DZIESIETNA PARAMETROW.

JEZYK SAS

Znaczenie:

Deklaracja ta stwierdza, że następujące po niej rozkazy są napisane w języku SAS^{*)}. Powrót do języka SAKO wymaga deklaracji JEZYK SAKO.

Uwaga:

W programie napisanym w języku SAS nie można użyć odpowiedników deklaracji ROZDZIAŁ, KONIEC oraz rozkazów IDZ DO ROZDZIAŁU.

JEZYK SAKO

Znaczenie:

Deklaracja ta stwierdza, że następujące po niej rozkazy są napisane w języku SAKO. Deklaracji tej używa się po uprzednim zastosowaniu deklaracji JEZYK SAS.

KONIEC

Znaczenie:

Deklaracja ta określa koniec napisanego programu.

^{*)} MASZYNA XYZ - Podręcznik programowania w języku maszyny. Systemy SAB, SAS i SO, Prace ZAM 1961 Ser. C Nr 1.

Uwagi:

Po wprowadzeniu deklaracji KONIEC maszyna czeka na wprowadzanie programu i w zależności od położenia kluczy na pulpicie sterującym maszyny:

- zatrzymuje się i po naciśnięciu przycisku START przechodzi do wykonywania programu,
- wyprowadza program w języku SAS,
- wyprowadza program w języku SAB,
- wyprowadza program w systemie oktalnym *)

3.2. Rozkazy sterująceSKOCZ DO α Przykłady:

SKOCZ DO 3

SKOCZ DO 5F

Znaczenie:

Rozkaz powoduje przejście do rozkazu o numerze α .

Forma:

α - numer rozkazu.

Uwaga:

Zamiast numeru α w rozkazie SKOCZ DO α można napisać słowo NASTEPNY; rozkaz SKOCZ powoduje wówczas przejście do rozkazu wypisanego w najbliższej kolejności /staje się rozkazem nieefektywnym/.

SKOCZ WEDLUG I : $\alpha, \beta, \dots, \omega$ Przykłady:

SKOCZ WEDLUG ALFA : 1, 2A, NASTEPNY

SKOCZ WEDLUG K2 : 3AB, 4, 2, 5C

*) w publikacji ZAM podanej na str. 52.

Znaczenie:

Rozkaz określa przejście do rozkazu o numerze, który występuje na liście w kolejności I. Jeśli zmienna I posiada wartość 0, rozkaz SKOCZ WEDŁUG I spowoduje przejście do rozkazu o numerze pierwszym na liście; gdy wartością I jest 2, wówczas jest wykonane przejście do rozkazu o numerze trzecim na liście itd.

Forma:

I - zmienna całkowita

$\alpha, \beta, \dots, \omega$ - numery rozkazów.

Każdy z tych numerów może być zastąpiony słowem NASTĘPNY.

Uwaga:

Przed wykonaniem rozkazu SKOCZ WEDŁUG I wartość zmiennej I powinna być przez program określona.

POWTORZ OD α : I = J(K)L

Przykłady:

POWTORZ OD 3 : ALFA = ^(!)29

POWTORZ OD 1ERF : C3 = 0.01(0.01)1.

POWTORZ OD 2 : AB1 = X0(DELTA)5

Znaczenie:

Rozkaz POWTORZ OD α : I = J(K)L powoduje wielokrotne wykonanie odcinka programu, zawartego między rozkazem oznaczonym numerem α , a samym rozkazem POWTORZ. Pierwszy raz dany fragment programu wykonuje się dla I = J, przy każdym następnym wykonaniu wartość I zwiększa się o K. Gdy zmienna I osiągnie wartość L, wówczas następuje przejście do następnego rozkazu po rozkazie POWTORZ.

Forma:

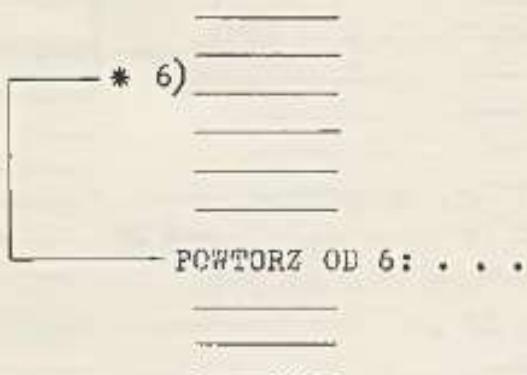
α - numer rozkazu

I, J, K, L - liczby lub zmienne proste.

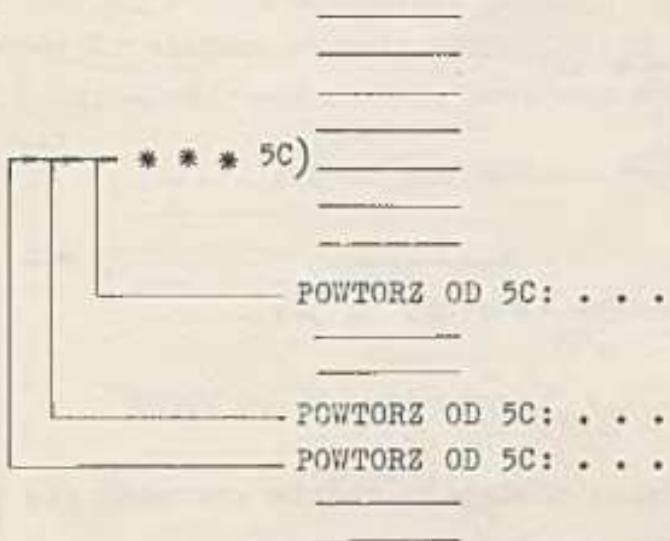
Ich wartości powinny posiadać jednakowy charakter, tzn. winny być jednocześnie całkowitymi lub ułamkowymi.

Reguły:

1. Stosując rozkaz POWTORZ OD α , stawiamy znak * z lewej strony numeru α :

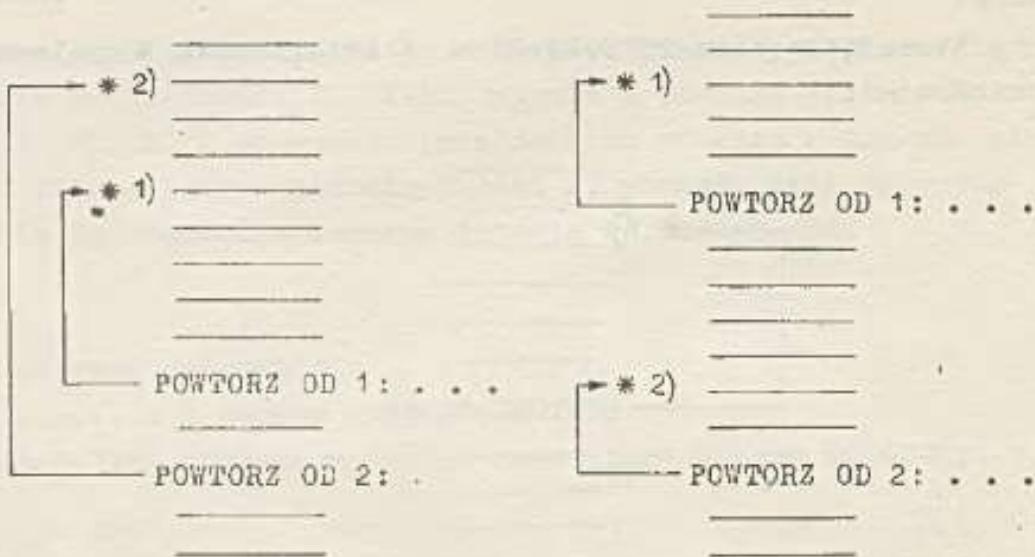


Jeżeli do jednego numeru α odnosi się kilka rozkazów POWTORZ, to przed tym numerem stawiamy tyle znaków *, ile jest tych rozkazów:

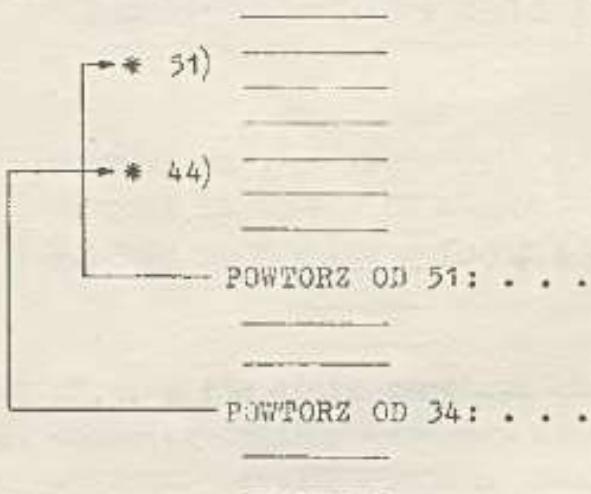


2. Rozkaz POWTORZ może się odnosić tylko do numeru rozkazu, wyisanego przed nim w programie.

3. Odcinek programu, zawarty między rozkazem o numerze α a rozkazem POWTORZ OD α nazywamy zasięgiem tego rozkazu POWTORZ. Dwa zasięgi mogą być albo całkowicie rozłączne /tzn. nie posiadać żadnego wspólnego rozkazu/, albo jeden z nich może zawierać się wewnątrz drugiego:

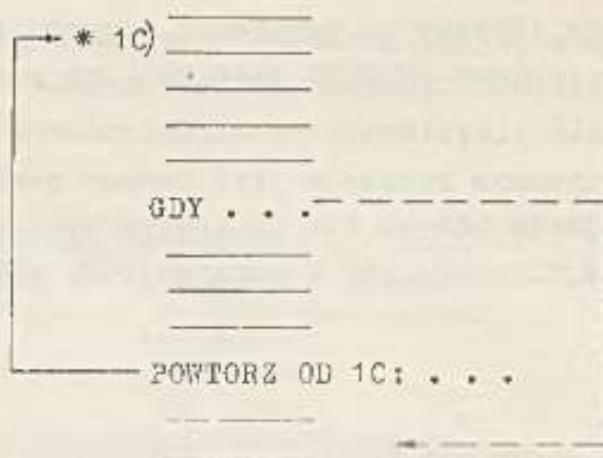


Nie jest prawidłowy więc następujący układ rozkazów POWТОRZ:



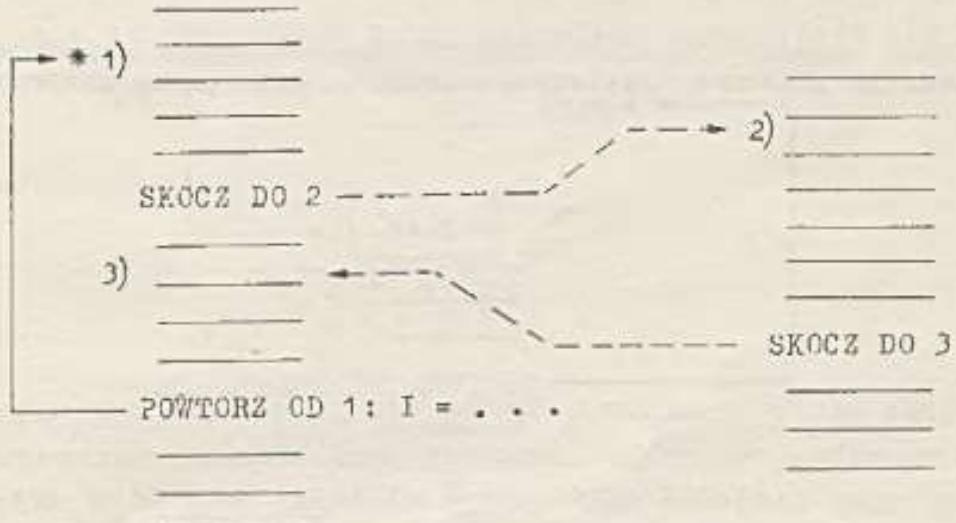
Zasadę tę wyrażamy krótko, mówiąc że "pętle nie mogą się przecinać".

4. Przerwanie powtarzania danego odcinka programu może nastąpić nie tylko dla wartości I równej L; w zasięgu rozkazu POWТОRZ może znajdować się rozkaz sterujący, który spowoduje przejście do wykonania rozkazu, położonego poza zasięgiem POWТОRZ. Mówimy wówczas, że działanie POWТОRZ zostało p r z e r w a n e:



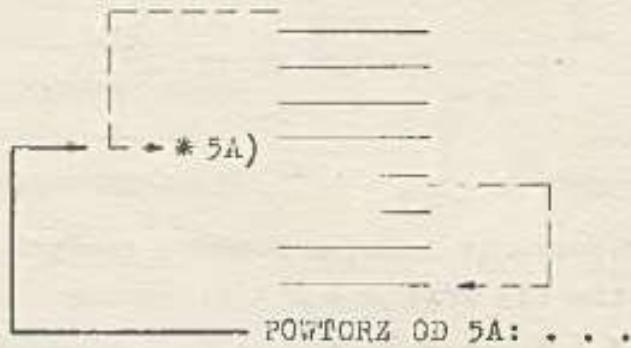
Jeśli natomiast powtarzanie zasięgu rozkazu powtórz zostało zakończone przez zrównanie się wartości I i L, mówimy, że działanie POWTORZ zostało wykonane.

5. Zarówno w przypadku, gdy rozkaz POWTORZ został przerwany jak i wykonany, ostatnia z przyjętych wartości I zostaje zachowana w dalszym ciągu programu. Umożliwia to również chwilowe przerwanie działania POWTORZ w celu wykonania pewnego odcinka programu, a następnie powrót do zakresu danego POWTORZ; ten odcinek programu może korzystać z aktualnej wartości I, lecz nie może zmieniać jej wartości;

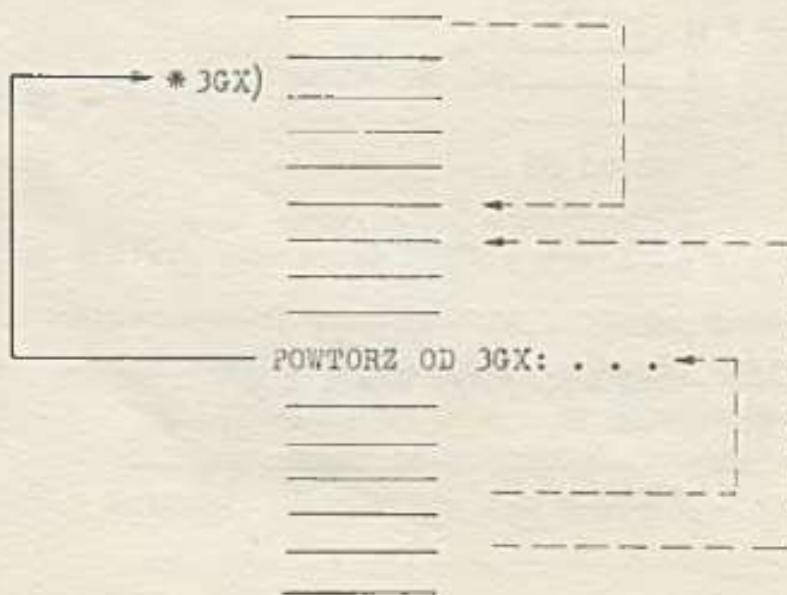


6. Przejście skokiem do wykonania rozkazu znajdującego się w zasięgu pewnego rozkazu POWTORZ jest możliwe ponadto w następujących przypadkach:

- gdy wykonywanym rozkazem jest rozkaz o numerze α,
- gdy przejście odbywa się na skutek wykonania rozkazu sterującego, znajdującego się w zasięgu tego samego rozkazu POWTORZ:



Poza opisanymi wyżej przypadkami, żadne przejście do wykonania rozkazu z zasięgu POWTORZ nie jest dopuszczalne. Niedopuszczalne są więc sytuacje, przedstawione na schemacie poniżej:



7. Jeśli J, K, L są ułamkowe i wartości ich są zaokrąglone tak, że wartość I nie może stać się równą wartości L dla żadnego powtórzenia, wówczas zakończenie powtarzania następuje dla wartości I najbliższej wartości L.

IDZ DO ROZDZIAŁU : nazwa

Przykłady:

IDZ DO ROZDZIAŁU: GAUSS SEIDEL

IDZ DO ROZDZIAŁU: LICZENIE PIERWIASTKOW

Znaczenie:

Rozkaz powoduje wczytanie z pamięci bęбnowej do pamięci wewnętrznej maszyny nowego rozdziału o podanej nazwie, a następnie przejście do wykonania pierwszego rozkazu z nowo wczytanego rozdziału.

Forma:

Nazwa rozdziału: ciąg liter, cyfr i spacji, rozpoczynający się od litery. Własność identyfikującą posiada 10 pierwszych znaków /nie licząc spacji/.

Uwaga:

Po wczytaniu nowego rozdziału wszelkie deklaracje zmienne, numery i nazwy podprogramów, używane w starym rozdziale, przestają być aktualne.

WROC

Znaczenie:

Rozkaz ten powoduje przejście od wykonywania rozkazów podprogramu do wykonywania rozkazów tego programu, który ten podprogram wywołał. Rozkaz WROC jest ostatnim wykonywanym rozkazem podprogramu. Przejście następuje do tego miejsca programu wywołującego, w którym nastąpiło wywołanie podprogramu. Mówiąc krótko, rozkaz WROC realizuje powrót z programu podległego do programu nadziednego.

GDY $A > B : \alpha$, INACZEJ β

Przykłady:

GDY $A \times X + B > 0 : 1A$, INACZEJ 4B

GDY $0 > X : 81$, INACZEJ NASTEPNY

GDY $B(I, J) > C(I) : NASTEPNY$, INACZEJ 2

Znaczenie:

Gdy podana nierówność jest spełniona, rozkaz powoduje przejście do rozkazu o numerze α ; w przeciwnym przypadku do rozkazu o numerze β .

Jeżeli zamiast α lub β występuje słowo NASTEPNY, oznacza ono rozkaz wypisany w najbliższej kolejności.

Forma:

A, B - wyrażenia arytmetyczne

α, β - numery rozkazów lub słowa NASTEPNY.

Uwaga:

Wartości A i B mogą mieć ten sam lub różny charakter, tzn. jedno z nich może mieć wartość całkowitą, zaś drugie ułamkową.

GDY $A = B : \alpha$, INACZEJ β

Przykłady:

GDY $A = B1 : 5$, INACZEJ 2

GDY $ALFA(I, K + 4) = 0 : NASTEPNY$, INACZEJ 4C

Znaczenie:

Gdy podana równość jest spełniona, rozkaz powoduje przejście do rozkazu o numerze α ; w przeciwnym przypadku do rozkazu o numerze β .

Jeżeli zamiast α lub β występuje słowo NASTEPNY, oznacza ono rozkaz wypisany w najbliższej kolejności.

Forma:

A, B - wyrażenia arytmetyczne
 α, β - numery rozkazów lub słowa NASTEPNY

Uwaga:

Patrz rozkaz GDY A>B.

GDY BYŁ NADMIAR: α , INACZEJ β

Przykłady:

GDY BYŁ NADMIAR: 1A, INACZEJ 3

GDY BYŁ NADMIAR: 4, INACZEJ NASTEPNY

Znaczenie:

Gdy we wskaźniku nadmiaru znajduje się liczba 1, rozkaz ten powoduje przejście do rozkazu o numerze α ; w przeciwnym przypadku /gdy we wskaźniku nadmiaru znajduje się liczba 0/ do rozkazu o numerze β .

Jeśli zamiast α lub β występuje słowo NASTEPNY, oznacza ono rozkaz wypisany w najbliższej kolejności.

Równocześnie rozkaz ten powoduje wpisanie liczby 0 do wskaźnika nadmiaru.

Forma:

α, β - numery rozkazów. Jeden z nich /lub oba/ może być zastąpiony słowem NASTEPNY.

Uwaga:

Pojawienie się liczby 1 we wskaźniku nadmiaru jest spowodowane wystąpieniem w maszynie liczby, przekraczającej zakres przyjętej skali /patrz LICZBA CAŁKOWITA, LICZBA UŁAMKOWA/.

GDY KLUCZ I : α , INACZEJ β
--

Przykłady:

GDY KLUCZ 15: 2CD, INACZEJ 3

GDY KLUCZ I : NASTEPNY, INACZEJ 5X

Znaczenie:

Gdy klucz, wskazany przez rozkaz, jest podniesiony w trakcie działania maszyny, rozkaz ten powoduje przejście do rozkazu o numerze α ; w przeciwnym przypadku do rozkazu o numerze β .

Jeśli zamiast α lub β występuje słowo NASTEPNY, oznacza ono rozkaz wypisany w najbliższej kolejności.

Forma:

I - zmienna prosta lub liczba całkowita przyjmująca wartości
0, 1, 2, ..., 35.

α, β - numery rozkazów, lub słowa NASTEPNY.

STOP α

Przykłady:

STOP 10

STOP 1ABC

Znaczenie:

Rozkaz powoduje zatrzymanie pracy maszyny. Po ponownym naciśnięciu przycisku START maszyna rozpoczyna pracę od rozkazu o numerze α .

Forma:

α - numer rozkazu lub słowo NASTEPNY

Uwaga:

Gdy zamiast numeru α w rozkazie STOP α występuje słowo NASTEPNY, oznacza ono rozkaz wypisany w najbliższej kolejności.

3.3. Rozkazy arytmetyczne i bulowskie

A = B

/formuła arytmetyczna/

Przykłady:

$$\begin{aligned} A &= B + C \\ X1 &= ALFA + LOG(SIN(X)) \\ I &= I + 1 \\ S &= S + A(I, J) * B(J, K) \\ D &= 45.6789 \end{aligned}$$
Znaczenie:

Rozkaz ten powoduje obliczenie wartości wyrażenia arytmetycznego B oraz nadanie tej wartości zmiennej A, stojącej po lewej stronie znaku =. Formuła arytmetyczna nadaje lub zmienia dotychczasowe wartości zmiennych.

Forma:

A - zmienna ułamkowa lub całkowita, prosta lub indeksowana,
 B - wyrażenie arytmetyczne.

Formuła arytmetyczna nie może zajmować więcej, niż jeden wiersz formularza.

Reguły:

1. Jeśli zmienna A jest całkowita, zaś wyrażenie B jest ułamkowe, wówczas zmiennej A nadaje się wartość wyrażenia B, zaokrągloną do liczby całkowitej.

2. Jeśli zmienna A jest ułamkowa, zaś wyrażenie B jest całkowite, wówczas zmiennej A nadaje się wartość B sprowadzoną do postaci ułamkowej, zgodnie z aktualnie ustaloną skalą.

Uwaga:

Znak = , występujący w formule, różni się znaczeniem od ogólnie przyjętego w matematyce; w formule określa on nie równość, lecz zmianę dotychczasowej wartości zmiennej.

A ≡ B

/formuła bulowska/

Przykłady:

A ≡ B + C

SDF ≡ 0000.4567.3241 * A

D ≡ D * (-23)

Znaczenie:

Rozkaz ten powoduje obliczenie wartości wyrażenia bulowskiego B i nadanie obliczonej wartości zmiennej A, stojącej po lewej stronie znaku ≡. Formuła bulowska nadaje lub zmienia dotyczasowe wartości zmiennych.

Forma:

A - zmienna ułamkowa lub całkowita /z reguły prosta/

B - wyrażenie bulowskie

Formuła bulowska nie może zajmować więcej niż jeden wiersz formularza.

Uwagi:

1. Znak ≡, występujący w formule bulowskiej, różni się znaczeniem od ogólnie przyjętego w matematyce; w formule określa on nie równość, lecz zmianę dotyczasowej wartości zmiennej.
2. Ponieważ słowo bulowskie może być traktowane jako liczba /ułamkowa lub całkowita/, zatem za pośrednictwem formuł bulowskich można nadawać wartości zmiennym, używanym w dalszym ciągu jako symbole liczb w wyrażeniach arytmetycznych i na odwrót: zmienne, których wartości były określone formułami arytmetycznymi, mogą w wyrażeniu bulowskim być traktowane jako symbole słów bulowskich.

(lista wyników) = nazwa (lista argumentów)

/formuła operacyjna/

Przykłady:

(X, ALFA, C1) = TRY X + Y, 7.89 - X + Y, SIN(X), R MALE

(*Y) = KROK RK(*Y, PRAWE STR(), 0.00001, 5)

(*A) = MN MACIERZY(*B, *C, 5, 6, 7)

(DF, T) = GX6(·, ·, ·, ·, ·, 45.6789)

Znaczenie:

Rozkaz ten określa wartości argumentów danego podprogramu, powoduje jego wykonanie i otrzymane rezultaty przypisuje zmiennym i blokom, występującym na liście wyników.

Formuła operacyjna stanowi uogólnienie formuły arytmetycznej; ta ostatnia nadaje wartości pojedynczej zmiennej przez wykonanie określonych działań na zmiennych wyrażenia arytmetycznego, podczas gdy formuła operacyjna nadaje wartości układowi zmiennych poprzez wykonanie działań podprogramu na układzie jego argumentów. W niektórych argumentów, będących liczbami, można podstawić wartości wyrażeń, których wyniki są zgodne co do charakteru z odpowiednimi argumentami podprogramu.

Forma:

Charakter zmiennych i ich kolejność na listach formuły operacyjnej muszą być zgodne z charakterem i kolejnością symboli, występujących na odpowiednich listach w deklaracji PODPROGRAM.

Reguła:

W chwili wykonywania formuły operacyjnej wszystkie argumenty wywoływanego podprogramu muszą być określone. Tym niemniej formuła operacyjna może określać przez podstawienie tylko niektóre z nich – pozostałe muszą być w tym przypadku podstawione uprzednio przez rozkaz PODSTAW. Na liście argumentów formuły wypisuje się wówczas tylko te zmienne, które ulegają podstawieniu. W pozostałych miejscach wpisuje się kropki.

Uwagi:

1. Nie jest konieczne wypisywanie kropek za ostatnim podstawianym argumentem. Tak więc zamiast pisać
$$(A, B) = TRAP(C, D, \cdot, \cdot, \cdot, \cdot, \cdot)$$
można krócej pisać:
$$(A, B) = TRAP(C, D)$$
2. Podstawione argumenty zachowują swoją wartość aż do wykonania kolejnego następnego podstawienia. Tak więc zmiana jednego argumentu w podprogramie nie wymaga dokonania wszystkich pozostałych podstawień.

PODSTAW: nazwa (lista)

Przykłady:

PODSTAW: TRY (A, B, C)

PODSTAW: TRANS (·, ·, S8)

PODSTAW: CALKA (·, F())

PODSTAW: ILMA (, ·, * A, 5)

Znaczenie:

Rozkaz ten powoduje określenie wartości argumentów podprogramu o podanej nazwie. Inaczej mówiąc, rozkaz ten powoduje podstawienie pewnych wielkości w miejsce odpowiednich argumentów podprogramem.

Forma:

1. Na liście wypisuje się tylko te wielkości, które ulegają podstawieniu. W miejscach przeznaczonych dla pozostałych argumentów /nie podstawianych tym rozkazem/ stawia się kropki, zachowując rozdzielające je przecinki. Kropki tych można nie stawiać po ostatniej wypisanej na liście wielkości.

2. Nazwy funkcji, podstawianych do podprogramu, uzupełnia się parą nawiasów, postawionych za nazwą funkcji; natomiast nazwy podstawianych bloków poprzedza się, zgodnie z ogólną regułą, symbolem *.

Reguły:

1. Charakter podstawianych wielkości musi być zgodny z charakterem argumentów, zadeklarowanych deklaracją PODPROGRAM.

2. Podstawione wielkości pozostają tak dugo argumentami podprogramu, aż nie nastąpi nowe podstawienie w miejsce tych samych argumentów. Może to być dokonane za pośrednictwem innego rozkazu PODSTAW bądź też formuły operacyjnej.

USTAW SKALE DZIESIETNIE: IPrzykłady:

USTAW SKALE DZIESIETNIE: 3

USTAW SKALE DZIESIETNIE: ALFA

Znaczenie:

Rozkaz ten ustala skalę dziesiętną wszystkich zmiennych ułamkowych, prostych i indeksowanych w rozkazach o dalszej kolejności wykonania – aż do nowego rozkazu ustalającego skalę.

Forma:

I – zmienna prosta całkowita lub liczba całkowita o wartości
0, 1, ..., 10

Reguła:

W każdym programie wykonanie rozkazów zawierających zmienne ułamkowe, musi być poprzedzone rozkazem USTAW SKALE DZIESIETNIE lub USTAW SKALE BINARNIE. Ustalona skala zachowuje swoje znaczenie przy przejściu do podprogramów oraz przy przejściu z jednego rozdziału do drugiego.

USTAW SKALE BINARNIE: IPrzykłady:

USTAW SKALE BINARNIE: 5

USTAW SKALE BINARNIE: ALFA

Znaczenie:

Rozkaz ten ustala skalę dwójkową wszystkich ułamków binarnych w rozkazach o dalszej kolejności wykonania – aż do nowego rozkazu ustalającego skalę.

Forma:

I – zmienna prosta lub liczba całkowita o wartości 0, 1, ..., 35.

Reguła:

Patrz rozkaz USTAW SKALE DZIESIETNIE

ZWIĘKSZ SKALE DZIESIETNIE O I: lista

Przykłady:

ZWIĘKSZ SKALE DZIESIETNIE O ABC : A, *B, C, *D

ZWIĘKSZ SKALE DZIESIETNIE O -7 : EF, *G

Znaczenie:

Rozkaz ten powoduje zwiększenie skali dziesiętnej o wielkości I zmiennych i bloków, wymienionych na liście.

Forma:

I - zmienna prosta lub liczba całkowita o wartości
-10, -9, -8, -7, ..., 0, 1, ..., +10

lista - zbudowana ze zmiennych prostych i nazw bloków.

Reguła:

Rozkaz ten można stosować jedynie wówczas, gdy wielkości, których nazwy występują na liście, są zapisane w maszynie w skali dziesiętnej /przy pomocy rozkazu USTAW SKALE DZIESIETNIE/.

Uwagi:

1. W przypadku zmiany ze skali większej na mniejszą ($I < 0$) może wystąpić nadmiar wraz z wszystkimi jego konsekwencjami.

Przykład:

A :	+ 0 0 3 2 . 8 5 6 2 8 0
	0 1 2 3 4 5 6 7 8 9 10

Rozkaz: ZWIĘKSZ SKALE DZIESIETNIE O 3:A powoduje powstanie nadmiaru i stratę najbardziej znaczących cyfr wartości zmiennej A.

2. W przypadku zmiany ze skali mniejszej na większą ($I > 0$) może nastąpić strata najmniej znaczących cyfr wartości zmiennych, których dotyczy ten rozkaz. Wartości tych zmiennych zostają wówczas zaokrąglone.

Przykład:

A :	+ 0 0 3 2 . 8 5 6 2 8 0
	0 1 2 3 4 5 6 7 8 9 10

Rozkaz: ZWIEKSZ SKALE DZIESIEINIE 0 2 : A powoduje następującą zmianę zapisu wartości A:

A :	+	0	0	0	0	3	2	.	8	5	6	3
		0	1	2	3	4	5	6	7	8	9	10

↑
zaokrąglenie

ZWIEKSZ SKALE BINARNIE o I : lista

Przykłady:

ZWIEKSZ SKALE BINARNIE 0 KL1 : A, * B, C, * D

ZWIEKSZ SKALE BINARNIE 0 -8 : E, F, * G

Znaczenie:

Rozkaz ten powoduje zwiększenie skali binarnej zmiennych i bloków, wypisanych na liście, o I.

Forma:

I - zmienna prosta całkowita lub liczba całkowita o wartości
 $-35, -34, \dots, +34, +35$

Uwaga:

Analogicznie, jak w przypadku ZWIEKSZ SKALE DZIESIETNIE.

3.4. Rozkazy wejścia i wyjścia

CZYTAJ : lista

Przykłady:

CZYTAJ: AB, * ALFA, OMEGA

CZYTAJ: * AKY1

Znaczenie:

Rozkaz powoduje wczytanie /w ramach pracy programu/ układu liczb z urządzenia wejściowego do pamięci wewnętrznej maszyny. Ponadto rozkaz ten nadaje wartości zmiennym z listy w ten sposób, że kolejnej zmiennej /lub blokowi/ przypisuje się kolejną liczbę /lub układ liczb/ z urządzenia wejściowego.

Forma:

lista - zbudowana jest ze zmiennych prostych i nazw bloków.

Reguły:

1. Każda liczba, przewidziana jako wartość zmiennej prostej, jak każdy układ liczb, stanowiący blok, musi zaczynać się od nowego wiersza formularza danych wejściowych.
2. Po ostatniej liczbie zapisanej w bloku następować musi oddzielny wiersz z wypisanym symbolem * .
3. Ilość liczb, stanowiących w programie blok, musi być dokładnie równa ilości przewidzianej przez właściwą deklarację BLOK lub STRUKTURA - w przeciwnym przypadku maszyna przerwie obliczenia, wykazując błąd.
4. Kolejność wypisania zmiennych na liście rozkazu CZYTAJ musi być zgodna z kolejnością wypisania danych na formularzu wejściowym.
5. Jeśli układ liczb na formularzu wejściowym stanowi blok, wówczas w jednym wierszu formularza można pisać kilka z tych liczb, oddzielając je spacjami. W związku z tym, między znakami jednej liczby nie może występować spacja. Kolejność wczytywania liczb, wypisanych w jednym wierszu, jest od lewej do prawej strony formularza.

*) Znaki liczby: + lub - , cyfry, kropka pozycyjna

6. Poszczególne liczby, wiersze lub bloki mogą być opatrzone komentarzami, nie wczytywanymi do pamięci maszyny. Komentarz musi zaczynać się od litery, a kończyć na znaku = lub : . Wśród znaków komentarza nie może występować ani znak = ani znak : . Komentarz może zajmować część, cały, a nawet kilka wierszy formularza.

7. Dane wejściowe wczytywane są do pamięci maszyny w skali określonej przez ostatnio wykonany rozkaz USTAW SKALE DZIESIETNIE lub USTAW SKALE BINARNIE. Jeżeli pewna wielkość wczytywana nie mieści się w przewidzianej skali, maszyna przerywa obliczenia wykazując błąd.

Uwaga:

Ilość liczb wczytywanych do pamięci maszyny jest równa sumie ilości elementów bloków i ilości zmiennych prostych, występujących na liście rozkazu CZYTAJ. Tak więc, jeśli A i B są zmiennymi prostymi, zaś C nazwą bloku o 25 elementach, rozkaz

CZYTAJ: A, B, C

spowoduje wczytanie

1 + 1 + 25 = 27

liczb do pamięci maszyny.

CZYTAJ WIERSZ : nazwa bloku

Przykłady:

CZYTAJ WIERSZ: ALFA

CZYTAJ WIERSZ: N

Znaczenie:

Rozkaz ten powoduje wczytanie do pamięci wewnętrznej maszyny następujących po sobie znaków, tworzących dane wejściowe, począwszy od skrajnie lewego znaku najbliższego wiersza. Każdy znak wiersza, wyrażony za pośrednictwem sześciu zer i jedynek, wpisywany jest do ostatnich sześciu pozycji każdego z kolejnych słów bloku o podanej nazwie. Znaki są wczytywane do znaku P.K. /Powrót Karetki/ włącznie.

Forma:

Na liście może występować tylko jedna nazwa bloku zadeklarowana uprzednio jako całkowita.

Reguła:

Znaki poprzedzone na taśmie symbolem "Litery" aż do pierwszego kolejnego znaku "Cyfry" wyłącznie uzupełniane są w trakcie czytania dodatkową jedynką na dwunastej pozycji Słowa /patrz zapis binarny liczb w maszynie, punkt 1.4.2. str. 17/.

Uwaga:

Ostatnim znakiem wczytywanego wiersza jest zawsze Powrót Karetki.

CZYTAJ OKTALNIE : lista

Przykłady:

CZYTAJ OKTALNIE : AB

CZYTAJ OKTALNIE : * ALFA

CZYTAJ OKTALNIE : AB, * ALFA

Znaczenie:

Rozkaz powoduje wczytanie słów bulowskich z urządzenia wejściowego do pamięci wewnętrznej maszyny z jednoczesnym ich przypisaniem kolejnym zmiennym na liście.

Forma:

Na liście występuwać mogą zmienne proste i nazwy bloków.

Reguła:

Wszystkie reguły rozkazu CZYTAJ odnoszą się do rozkazu CZYTAJ OKTALNIE, z tym że przez liczbę rozumiemy słowo bulowskie.

DRUKUJ (I,J) : lista

Przykłady:

DRUKUJ (2,5) : AX, DELTA

DRUKUJ (P,9) : A(K+1), A(K+2), A(K+3), A(K+4)

DRUKUJ (12) : BCD 9

DRUKUJ (3) : EFG I

Znaczenie:

Rozkaz powoduje kolejne wypisanie w aktualnie wypisanym wierszu wartości zmiennych podanych na liście. Parametry rozkazu określają ilość miejsc zarezerwowanych dla wypisania wartości każdej zmiennej oraz sposób jej wypisania.

Forma:

lista - składa się z

1. zmiennych prostych,
2. zmiennych indeksowanych o jednym indeksie będącym liczbą lub zmienną,
3. zmiennych indeksowanych, których indeksem jest suma zmiennej i liczby;

parametry -

1. - I,J - zmienne całkowite lub liczby całkowite.

2. gdy drukowanie dotyczy liczb ułamkowych,

I oznacza ilość zarezerwowanych pozycji przed kropką dziesiętną, zaś J ilość pozycji za kropką. Łącznie z kropką i znakiem, powyższy rozkaz powoduje więc zarezerwowanie I+J+2 pozycji arkuszy wydawniczego na wypisanie każdej liczby,

3. gdy drukowanie dotyczy liczb całkowitych, rozkaz ma postać

DRUKUJ I : lista

gdzie I oznacza ilość zarezerwowanych pozycji na wypisanie liczby całkowitej. Łącznie z miejscem na ewentualny znak powyższy rozkaz powoduje zarezerwowanie I+1 pozycji arkusza wydawniczego

Reguły:

1. W przypadku liczb ułamkowych sposób drukowania jest następujący:

- kropkę dziesiętną wypisuje się w I+2 zarezerwowanej pozycji,
- część całkowitą wypisuje się przed kropką. Bezpośrednio przed liczbą dodatnią pisze się znak +, przed ujemną znak -. Początkowe zarezerwowane pozycje, nie zajęte cyframi znaczącymi liczby i znakiem liczby, wypełniane są spacjami. Gdy żądamy zero cyfr przed kropką, to znak wystąpi tuż przed kropką

-.7341

gdy żądamy większej ilości cyfr przed kropką i część całkowita jest zero, to między znakiem a kropką zostanie napisana cyfra "0"

-0.7341 .

2. W przypadku liczb całkowitych liczbę drukuje się bez kropki pozycyjnej, tak że ostatnia cyfra liczby znajdowała się w I+1 zarezerwowanej przez rozkaz pozycji. W liczbach dodatnich opuszcza się znak +, w ujemnych znak stawiany jest bezpośrednio przed pierwszą znaczącą cyfrą liczby. Początkowe zarezerwowane pozycje, nie zajęte cyframi znaczącymi oraz znakiem -, wypełniane są spacjami. W wypadku gdy liczba jest zerem, piszemy 0 lub -0 w zależności od znaku.

3. Skala drukowanych liczb ułamkowych jest zgodna z ostatnio wykonanym rozkazem USTAW SKALE.

4. W przypadku zbyt małej ilości miejsc zarezerwowanych na wypisanie liczby maszyna się zatrzymuje, sygnalizując błąd.

Uwaga:

Należy uważać, aby rozkazy DRUKUJ wraz z rozkazami TEKST i SPACJA nie przekroczyły pojemności jednego wiersza, która wynosi 69 znaków.

DRUKUJ WIERSZ: nazwa bloku

Przykłady:

DRUKUJ WIERSZ : ALFA

DRUKUJ WIERSZ : M

Znaczenie:

Rozkaz ten powoduje wypisanie wiersza począwszy od skrajnie lewej pozycji arkusza w ten sposób, że każdy kolejny znak dalekopiowy jest określony przez sześć najmniej znaczących pozycji kolejnego słowa bloku o podanej nazwie. Zakończenie drukowania następuje ze znakiem P.K. /Powrót Karetki/ włącznie.

Forma:

W liście może występować tylko jedna nazwa bloku, zadeklarowana wcześniej jako całkowita.

Reguła:

Ilość słów w bloku, a tym samym ilość wypisanych znaków dalekopiowych nie może przekroczać 69.

DRUKUJ OKTALNIE: lista zmiennych

Przykłady:

DRUKUJ OKTALNIE: A, B, C

DRUKUJ OKTALNIE: D(I)

Znaczenie:

Rozkaz powoduje wypisanie wartości zmiennych wypisanych na liście, traktując je jako słowa bulowskie /patrz słowo bulowskie/.

Forma:

lista - jest zbudowana ze zmiennych prostych bądź indeksowanych o jednym indeksie, będącym liczbą bądź zmienną.

Reguła:

Słowo jest drukowane w formie oktalowej przy użyciu 6 lub 12 cyfr: 0, 1, 2, 3, 4, 5, 6, 7 oraz kropek, stawianych co 3 znaki. Łącznie, na wydrukowanie słowa zarezerwowanych jest 15 pozycji.

W przypadku gdy zmieniona występująca na liście jest zadeklarowana jako całkowita na wydrukowanie słowa jest zarezerwowane 7 pozycji.

TEKST:

/dane tekstu/

Przykłady:

TEKST:

PIERWIASTEK =

TEKST:

TEMPERATURA OTOCZENIA WYNOSI:

Znaczenie:

Rozkaz powoduje kolejne wypisanie danych tekstu w aktualnie drukowanym wierszu.

Forma:

Dane tekstu wypisujemy w następnym wierszu po słowie TEKST. Początkiem danych tekstu jest pierwszy znak, nie będący spacją; końcem – ostatni znak nie będący spacją.

Uwaga:

Wewnątrz danych tekstu spacje zachowują swe znaczenie.

TEKST WIERSZY n:
/n wierszy tekstu/

Przykłady:

TEKST WIERSZY 2:
ROZWIAZANIE UKŁADU ROWNAN LINIOWYCH
METODA ITERACJI

TEKST WIERSZY 1:
TABLICA WARTOSCI FUNKCJI GAMMA

Znaczenie:

Rozkaz powoduje przepisanie na formularz wyjściowy podanych n wierszy tekstu.

Forma:

n - liczba naturalna.

Reguły:

1. Przy przepisywaniu uwzględnione są wszystkie pozycje danych tekstu, wliczając spacje, od początku aż do końca wierszy.
2. Wiersze niezapisane liczą się tak samo, jak wiersze zapisane. Umożliwia to tworzenie odstępów między wierszami wypisywanego tekstu.

SPACJA n

Przykłady:

SPACJA 3
SPACJA

Znaczenie:

Rozkaz powoduje pozostawienie n spacji w aktualnie wypisywanym wierszu.

Forma:

n - liczba naturalna

Uwaga:

Zamiast pisać SPACJA i można krócej pisać SPACJA

LINIA n

Przykłady:

LINIA 5

LINIA 8

LINIA

Znaczenie:

Rozkaz powoduje przejście do wypisywania danych wyjściowych poczawszy od skrajnie lewej pozycji n-tego kolejnego wiersza formularza danych wyjściowych.

Forma:

n - liczba naturalna.

Reguły:

1. Rozkazy: DRUKUJ, SPACJA oraz TEKST powodują kolejne wypisywanie wyników w obrębie jednego wiersza. Przejście do nowego wiersza formularza wymaga zastosowania rozkazu LINIA i wykonujemy je wtedy, gdy wymaga tego żądana przez nas forma danych wyjściowych lub gdy zachodzi obawa przekroczenia dopuszczalnej /9/ ilości znaków, mieszczących się w jednym wierszu formularza.

2. Rozkaz LINIA n można napisać w formie LINII n.

3. Rozkaz LINIA 1 można krócej pisać: LINIA.

Uwaga:

Rozkaz LINIA n powoduje pozostawienie na formularzu wyjściowym n-1 wierszy wolnych.

3.5. Rozkazy współpracy z bębnem

CZYTAJ Z BEBNA OD I : lista

Przykłady:

CZYTAJ Z BEBNA OD 120: A, *BETA

CZYTAJ Z BEBNA OD 2134: X, A, -

R, B, *DZETA, KSI

CZYTAJ Z BEBNA OD KSI: *C, *D

Znaczenie:

Rozkaz powoduje przepisanie liczb z pamięci bębnowej do pamięci wewnętrznej maszyny traktując je jako wartości zmiennych i bloków, wymienionych na liście. Liczby te przepisuje się z bębna kolejno poczynając od miejsca na bębnie, oznaczonego numerem I. Kolejne liczby stanowią wartości kolejnych zmiennych i bloków w takim porządku, w jakim są one wypisane na liście.

Forma:

I - liczba naturalna lub zmienna prosta, całkowita o wartości dodatniej,

lista - jest utworzona ze zmiennych prostych i nazw bloków.

Uwagi:

1. Ilość liczb, wczytywanych z bębna do pamięci wewnętrznej maszyny, jest równa sumie ilości zmiennych prostych i ilości elementów bloków, wymienionych w liście.

Tak więc jeśli A jest nazwą bloku o 25 elementach B jest nazwą bloku o 60 elementach, zaś C jest zmienną prostą, wówczas rozkaz:

CZYTAJ Z BEBNA OD 120: A, B, C

powoduje wczytanie do pamięci wewnętrznej maszyny ogółem 86 liczb, zapisanych na bębnie w miejscach:

120	}	blok A
121		
.		
.	}	blok B
144		
145		
146	}	zmienna C
.		
.		
204	}	zmienna C
205		

2. Kolejność wypisania zmiennych na liście rozkazu CZYTAJ Z BEBNA musi być zgodna z kolejnością zapisu liczb na bębnie.

PISZ NA BEBEN OD I: lista

Przykłady:

PISZ NA BEBEN OD 345: AB, CDEF, *D4

PISZ NA BEBEN OD GHI: KLM

Znaczenie:

Rozkaz powoduje przepisanie wartości zmiennych, podanych na liście, z pamięci wewnętrznej maszyny do pamięci bębowej. Liczby te wpisuje się do pamięci bębowej kolejno, począwszy od numeru miejsca na bębnie I. Rozkaz ten można nazywać "odwrotnym" w stosunku do rozkazu CZYTAJ Z BEBNA.

Forma:

I - liczba naturalna lub zmienna prosta, całkowita o wartości do datniej,

lista - jest utworzona ze zmiennych prostych i nazw bloków.

Uwaga:

Analogiczna, jak w rozkazie CZYTAJ Z BEBNA.

3.6. KomentarzPrzykłady:

- K) PIERWIASTEK JEST UJEMNY
- K) PRZYPADEK, GDY X-GAMMA JEST > 0
- K) BADANIE, CZY WARUNEK ZBIEZ-
- K) NOSCI JEST SPELNIONY

Znaczenie:

Komentarz nie wpływa na działanie programu, posiada jedynie znaczenie objaśniające. Używa się go, aby uczynić program bardziej przejrzystym.

Forma:

Komentarz może składać się z kilku wierszy. Jeden wiersz komentarza stanowi ciąg dowolnych znaków dalekopisowych, którego pierwszym znakiem jest litera K, zaopatrzona z prawej strony w nawias zamkający.

4. PODPROGRAMY W JĘZYKU SAKO

Podprogramem nazywamy wyodrębniony fragment programu, służący do wykonywania pewnych ustalonych działań na przekazanych mu wielkościach. Ścisłe mówiąc, podprogram realizuje w maszynie operację typu:

$$(u, v, \dots, w) = f(x, y, \dots, z)$$

przekształcającą układ argumentów x, y, \dots, z na układ wyników u, v, \dots, w .

Podprogram stanowi zazwyczaj część programu wielokrotnie wykonywaną; przez wyodrębnienie jej w podprogram całość programu staje się krótsza i bardziej przejrzysta. Dla często spotykanych operacji tworzy się raz na zawsze ułożone podprogramy. Korzystanie z nich jest proste i nie wymaga każdorazowego programowania, lecz tylko dołączania uprzednio napisanych podprogramów do tzw. programu głównego.

Przygotowanie programów dla maszyny cyfrowej jest tym łatwiejsze, im więcej istnieje przygotowanych zawozaszu podprogramów i im łatwiejszy jest sposób ich wykorzystania w każdym z konkretnych przypadków. Z tego względu przy budowie języka SAKO szczególną uwagę zwrócono na system współpracy programu głównego z podprogramami.

4.1. Terminologia i oznaczenia

W punkcie tym opisane są pewne pojęcia i terminy, używane w dalszym ciągu.

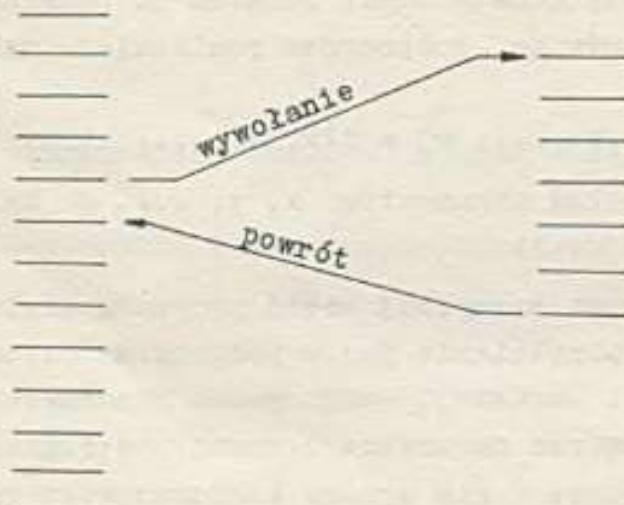
Przejście do wykonania podprogramu nazywamy **w y w o ł a - n i e m** podprogramu. W trakcie obliczeń jeden podprogram bywa zazwyczaj wywoływany kilkakrotnie.

Program, wywołujący podprogram, nazywa się **n a d r z e d - n y m** w stosunku do danego podprogramu; podprogram wywoływany nosi nazwę **p o d l e g ę c g o** w stosunku do programu nadziednego.

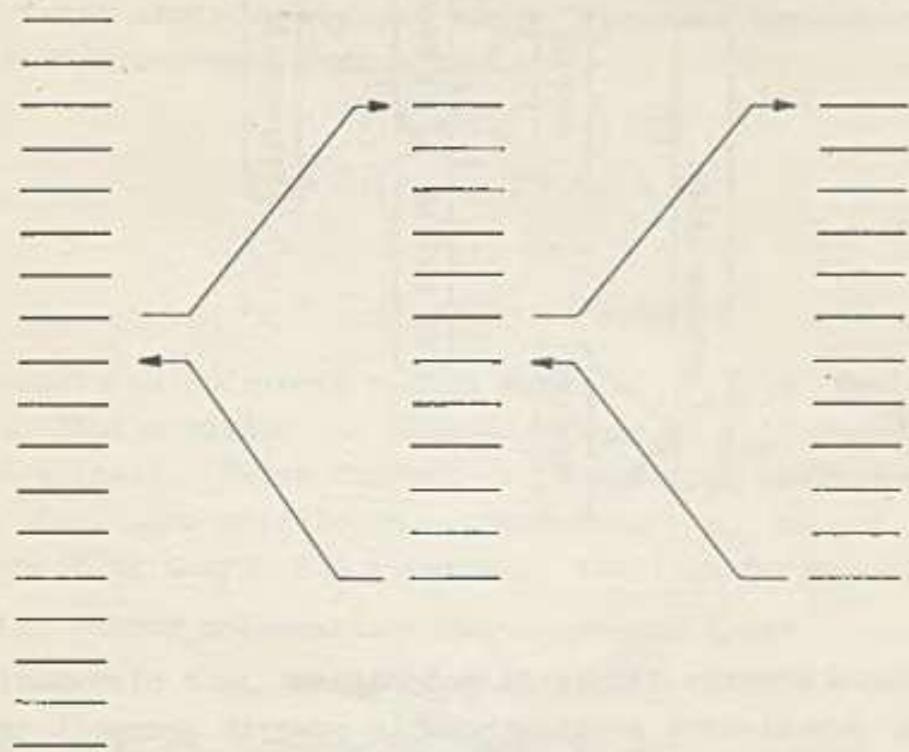
Przejście do wykonywania programu nadziednego nazywa się **p o - w r o t e m**.

nadrzędny

podległy



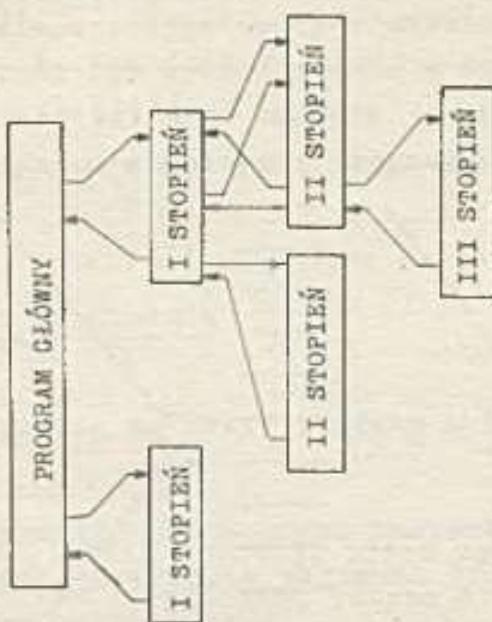
Program, nadzędny w stosunku do pewnego podprogramu może być sam podprogramem.



Zachodzi wówczas przypadek tzw. wielostopniowego korzystania z podprogramów.

Część programu, która nie jest podległa żadnej innej części, nosi nazwę programu głównego. Każdy rozdział programu pisanej w SAKO posiada dokładnie jeden program główny oraz ewentualnie pewną liczbę podprogramów.

Podprogramy, podległe programowi głównemu, nazywają się podprogramami I stopnia; podprogramy, podległe podprogramom I stopnia, noszą nazwę podprogramów II stopnia itd. O tym, czy podprogram jest podprogramem tego lub innego stopnia, decyduje sposób jego wykorzystania w konkretnym rozdziale programu. Ten sam podprogram może być w jednym rozdziale programu podprogramem stopnia I, zaś w drugim - podprogramem stopnia II. Poniżej podajemy schemat wielostopniowego korzystania z podprogramów:



Zasadniczym efektem działania podprogramu jest otrzymanie układu wyników, określonych przez wykonanie pewnych operacji arytmetycznych lub bulowskich na układzie argumentów podprogramu:

$$(u, v, \dots, w) = f(x, y, \dots, z)$$

wyniki argumenty

Operacją takiego typu jest na przykład przekształcenie współrzędnych przy przejściu z jednego układu odniesienia do drugiego; argumentami są stare współrzędne, zaś wynikami nowe, przekształcone.

Gdy wynikiem operacji jest jedna liczba, operacja ta nosi nazwę funkcji:

$$u = f(x, y, \dots, z)$$

Każdy podprogram posiada ścisłe określoną i ustaloną ilość, kolejność i charakter argumentów i wyników, wyrażonych przy pomocy symboli ogólnych. Aby wywołać podprogram, należy te symbole ogólne zastąpić symbolami o konkretnych znaczeniach, takich jak liczby, bloki itp. Proces zastępowania symboli ogólnych konkretnymi nosi nazwę przedstawienia.

Argumentami operacji mogą być w zasadzie dowolne obiekty matematyczne, tak jak liczby, wektory, liczby zespolone itp. W szczególności, argumentami mogą być inne operacje lub funkcje. Przykładem może tutaj służyć operacja, zwana "krokiem całkowania" układu równań różniczkowych zwyczajnych:

$$y'_0 = f_0(y_0, y_1, \dots, y_n)$$

$$y'_1 = f_1(y_0, y_1, \dots, y_n)$$

· · · · · · · · · ·

$$y'_n = f_n(y_0, y_1, \dots, y_n)$$

Operacja ta na podstawie układu wartości $y_i(t)$, kroku całkowania h , liczby równań n i układu funkcji $f_i()$ określa układ wartości $y_i(t+h)$. Układ funkcji $f_i()$ może być traktowany jako operacja $f()$, przekształcająca układ liczb y_i na układ liczb y'_i . Operację tę nazywa się zazwyczaj "obliczanie prawych stron".

Tak więc "krok całkowania" jest operacją typu:

$$(*z) = g(*y, h, n, f())$$

gdzie przez $*y$, $*z$ oznaczamy wektory rozwiązań w punktach t i $t+h$.

Aby wyróżnić spośród innych operacje tego typu, wprowadzamy następującą definicję:

1. Operacja, której żaden argument nie jest operacją, nosi nazwę operacji pierwszej rzędu.
2. Operacja, której jeden argument jest operacją rzędu N , zaś wśród pozostałych argumentów nie ma operacji o rzędzie większym od N , nosi nazwę operacji rzędu $N+1$.

W myśl tej definicji, przekształcenie układu współrzędnych jest operacją rzędu I, natomiast "krok całkowania" jest operacją rzędu II.

Podprogram, który definiuje operację rzędu N , nosi nazwę podprogramu rzędu N .

Operacja drugiego rzędu, której wynikiem jest jedna liczba, nazywa się zgodnie z terminologią matematyczną funkcjoną. Przykładem funkcjonału jest operacja obliczania całki z zadanej funkcji. Operacja ta na podstawie liczb a i b /granic całkowania/ liczby ϵ /zadanej dokładności obliczeń/ oraz funkcji podcałkowej $g()$ znajduje liczbę c :

$$c = \int_a^b g(x)dx + R \quad |R| < \epsilon$$

Operacja ta jest więc typu:

$$c = f(a, b, \epsilon, g()) .$$

Operacje rzędu II i wyższego często występują w problemach obliczeniowych. Narzuca to konieczność zbudowania takiego systemu korzystania z podprogramów, który by w łatwy sposób pozwalał stosować operacje wyższych rzędów.

4.2. Ogólne własności systemu korzystania z podprogramów w języku SAKO

Do cech charakterystycznych systemu SAKO, związanych z wykorzystaniem podprogramów należy

- możliwość tworzenia i korzystania z podprogramów dowolnie wysokiego rzędu,
- możliwość wielostopniowego korzystania z podprogramów, ograniczona jedynie pojemnością pamięci maszyny,
- niezależna numeracja rozkazów i niezależne stosowanie zmiennych w każdym podprogramie i programie głównym,
- możliwość wielokrotnego wywoływania podprogramu przez dowolny podprogram niższego stopnia,
- możliwość dokonywania podstawień w podprogramach przez dowolne podprogramy niższego stopnia,
- możliwość traktowania jako funkcji języka tych podprogramów, których językiem jest jedna liczba,

- możliwość używania jako argumentów podprogramu
 - liczb,
 - zmiennych,
 - bloków liczbowych,
 - operacji.
- możliwość otrzymywania w wyniku
 - liczb,
 - bloków liczbowych.

4.3. Budowa podprogramów SAKO

Podprogramy pisze się w standartowym języku SAKO. Program główny wraz z odpowiadającymi mu podprogramami musi mieścić się w jednym rozdziale programu. Podprogram rozpoczyna się zawsze od deklaracji

PODPROGRAM: (lista wyników) = nazwa (lista argumentów)

Deklaracja ta określa nazwę danego podprogramu oraz ilość, kolejność i charakter jego argumentów i wyników.

Dla większej przejrzystości oraz dla ułatwienia pracy translatora, wyróżnia się spośród argumentów i wyników podprogramów

- bloki liczbowe, poprzedzając ich nazwy symbolem *,
- operacje, przez dopisywanie po ich nazwach pary nawiasów () .

PODPROGRAM: (U, V) = TRANS(X, Y, ALFA)

PODPROGRAM: (* C) = MN MACIERZY (* A, * B, N, M, L)

PODPROGRAM: (* Z) = KROK RK (* Y, H, N, T, F())

Po deklaracji PODPROGRAM następują wszelkie inne deklaracje i rozkazy podprogramu. Występować w nich mogą tylko zmienne, wprowadzone przez rozkazy podprogramu, bądź też zmienne, wymienione w deklaracji PODPROGRAM. Tym ostatnim wartości nadaje program nadzędny przez podstawienie. Ostatnim wykonywanym rozkazem podprogramu jest zawsze rozkaz WROC, który realizuje powrót do programu nadzędnego. Tak więc jedyną komunikacją między programem nadzędnym a podległym są z jednej strony rozkazy PODSTAW oraz formuły arytmetyczne.

tyczne i operacyjne, które powodują podstawienia argumentów i wywo-żają program podległy /będzie o nich mowa w punkcie 4.4 str. 89/ z drugiej zaś strony rozkaz WROC.

W podprogramach stosuje się symbolikę niezależną od innych podprogramów i programu głównego. Oznacza to, że w podprogramie można używać nazw zmiennych i numerów rozkazów takich samych, jak w pozostałych częściach programu, mimo różnych znaczeń, jakie mogą one posiadać. Wynika stąd możliwość wielokrotnego wykorzystywania raz przygotowanego podprogramu; podprogram taki, posiadający niezależną symbolikę, można dołączać do każdego programu głównego. Również wszelkie deklaracje programu nadziednego /z wyjątkiem deklaracji dotyczących skali/ tracą moc w stosunku do programu podległego.

Rozkazy i deklaracje podprogramu mają identyczną formę i znacze-nie, jak rozkazy programu głównego. Poniżej przytaczamy przykład budowy prostego podprogramu:

```
PODPROGRAM: (U, V) = TRANS(X, Y, ALFA)
C = COS(ALFA)
S = SIN(ALFA)
U = X * C + Y * S
V = -X * S + Y * C
WROC
```

Jeśli operacja jest funkcją, tzn. posiada tylko jeden wynik liczbowy, wówczas forma budowy podprogramu, definiującego tę funk-cję ma nieco odmienną postać od formy innych podprogramów.

Deklaracja PODPROGRAM, odpowiadająca takiemu podprogramowi ma wówczas postać

```
PODPROGRAM: nazwa (lista argumentów),
zaś ostatnim wykonywanym rozkazem arytmetycznym takiego podprogra-mu jest rozkaz
nazwa funkcji = odp. wyrażenie arytmetyczne
lub:
nazwa funkcji ≡ odp. wyrażenie bulowskie .
```

Przykład:

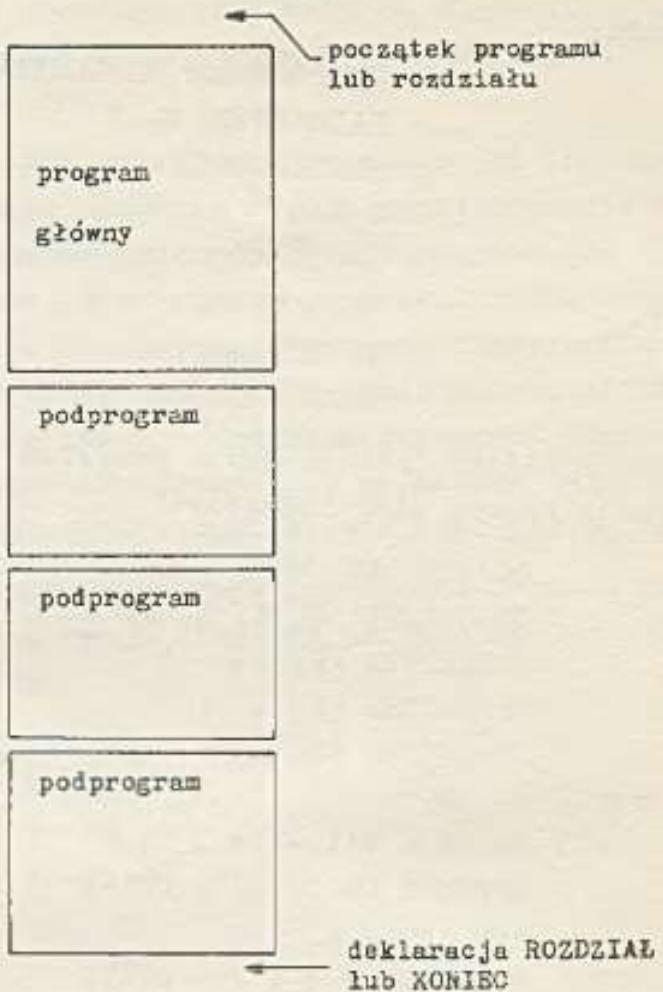
PODPROGRAM: WIELOMIAN (X, * A, N)
 CALKOWITE: N, I
 STRUKTURA (N) : A
 S = 0
 * 1) S = S × X + A(I)
 POWTORZ OD 1 : I = N(-1)0
 WIELOMIAN() = S
 WROC

Poniżej podajemy jeszcze jeden przykład budowy podprogramu, którego wyniki tworzą blok liczbowy:

PODPROGRAM: (* C) = MN MACIERZY (* A, * B, N, M, L)
 CALKOWITE: N, M, L, I, J, K
 STRUKTURA (N, M) : A
 STRUKTURA (M, L) : B
 STRUKTURA (N, L) : C
 * * 2) S = 0
 * 1) S = S + A(I, J) × B(J, K)
 POWTORZ OD 1 : J = 0(1)M
 C(I, K) = S
 POWTORZ OD 2 : I = 0(1)N
 POWTORZ OD 2 : K = 0(1)L
 WROC

4.4. Korzystanie z podprogramów, napisanych w języku SAKO

Podprogram, używany w określonym rozdziale programu, z reguły wchodzi w skład tego rozdziału. Rozdział składa się z programu oraz pewnej liczby podprogramów, wypisanych po programie głównym:



Korzystanie z podprogramu wymaga wykonania podstawień oraz wywołania podprogramu. Dokonuje się tego za pośrednictwem formuł arytmetycznych, jeśli podprogram definiuje funkcję, lub za pośrednictwem rozkazu

$$(u, v, \dots, w) = f(x, y, \dots, z)$$

zwanego w dalszym ciągu formułą operacyjną. Formuła operacyjna określa argumenty wywoływanego podprogramu, wyniki zaś wykonywanej operacji przyporządkowuje zmiennym, stojącym po lewej stronie znaku =.

Formuła operacyjna stanowi pewną analogię formuły arytmetycznej. Formuła arytmetyczna nadaje wartość pojedynczej zmiennej, podczas gdy formuła operacyjna nadaje wartości układowi zmiennych lub bloków. Formułę operacyjną można traktować jako pewną ilość formuł arytmetycznych, określających wartości zmiennych, stojących na liście wyników.

Poniżej dajemy przykład wykorzystania operacji TRANS za pośrednictwem formuły operacyjnej

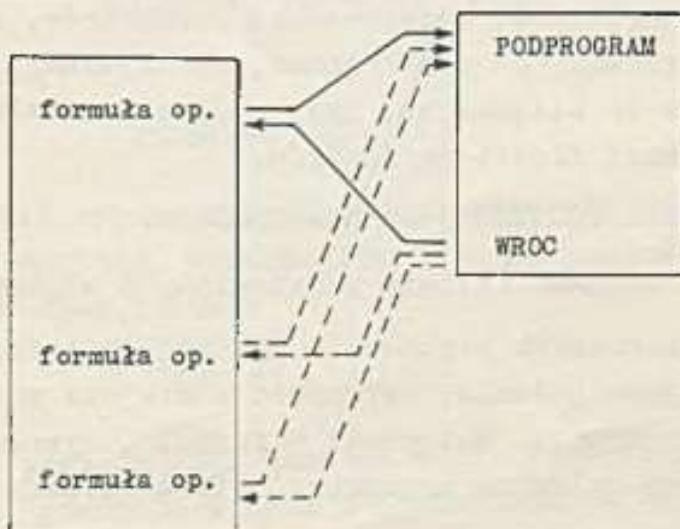
$(X, Y) = \text{TRANS}(X, 5.6789, \text{BETA})$

PODPROGRAM: $(U, V) = \text{TRANS}(X, Y, \text{ALFA})$

Kolejność i charakter symboli, stojących na listach formuły operacyjnej, musi być zgodna z kolejnością i charakterem symboli, wymienionych deklaracji PODPROGRAM.

Powrót do programu nadzawanego zabezpiecza rozkaz WRÓC, który powoduje przejście do wykonywania działań programu nadzawanego od tego miejsca poczawszy, w którym nastąpiło wywołanie podprogramu

program nadzawany



Jeśli podprogram definiuje funkcję, wówczas korzystamy z niego za pośrednictwem formuły arytmetycznej według takich samych zasad, jak z funkcji języka. Nazwa podprogramu jest w tym przypadku traktowana jako nazwa funkcji

Przykład:

A = B + C * J1(Y + H DELTA, 5)

PODPROGRAM: J1(X, N)

WROC

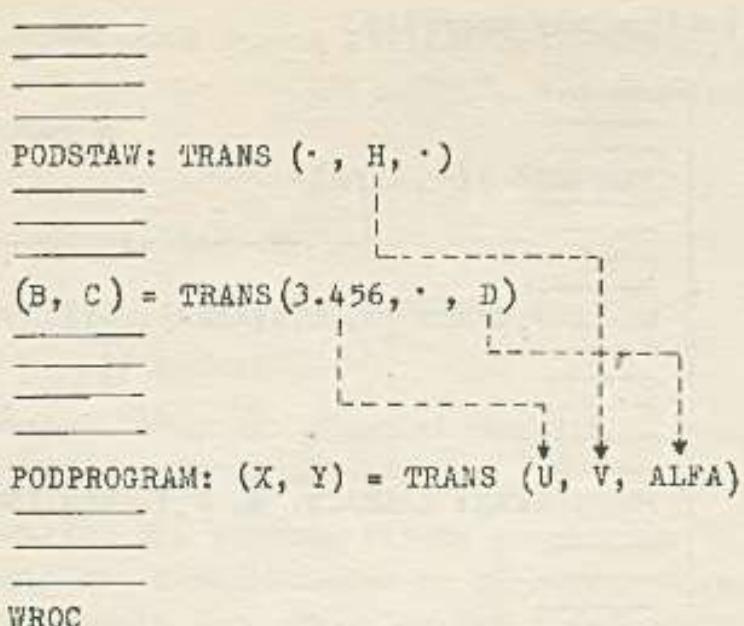
Stosowanie formuły arytmetycznej lub operacyjnej nie jest jedynym sposobem podstawienia konkretnych wartości do podprogramu. Część lub nawet wszystkie argumenty mogą być podstawione do podprogramu za pośrednictwem rozkazu PODSTAW.

Przyczyną wprowadzenia rozkazu PODSTAW jest uniezależnienie procesu podstawiania od procesu wywoływania podprogramu. Jest bowiem niezbędna możliwość podstawiania parametrów, obliczanych przez program główny, do podprogramu, wywoływanego przez inny podprogram, jak ma to miejsce np. przy obliczaniu całki z funkcji, zależnej od pewnej ilości parametrów.

Postać rozkazu PODSTAW jest następująca:

PODSTAW: f(lista podstawianych argumentów)

Na liście podstawianych argumentów wypisujemy tylko te argumenty, które ulegają podstawieniu, natomiast w miejsce pozostałych argumentów stawiamy kropki. Natomiast w formule, wywołującej podprogram, wypisujemy tylko te argumenty, które uprzednio nie były podstawiane:



WRÓC

Na powyższym rysunku kreską przerywaną zaznaczono drogi podstawień.

Kropki służą jedynie do określania właściwej kolejności podstawianych argumentów, tzn. ilość kropek na liście, poprzedzających dany argument wyznacza jego kolejność wśród argumentów. Nie jest konieczne wypisywanie kropek za ostatnim podstawianym argumentem. Zamiast więc pisać:

PODSTAW TRANS (A, . , .)

można pisać krócej:

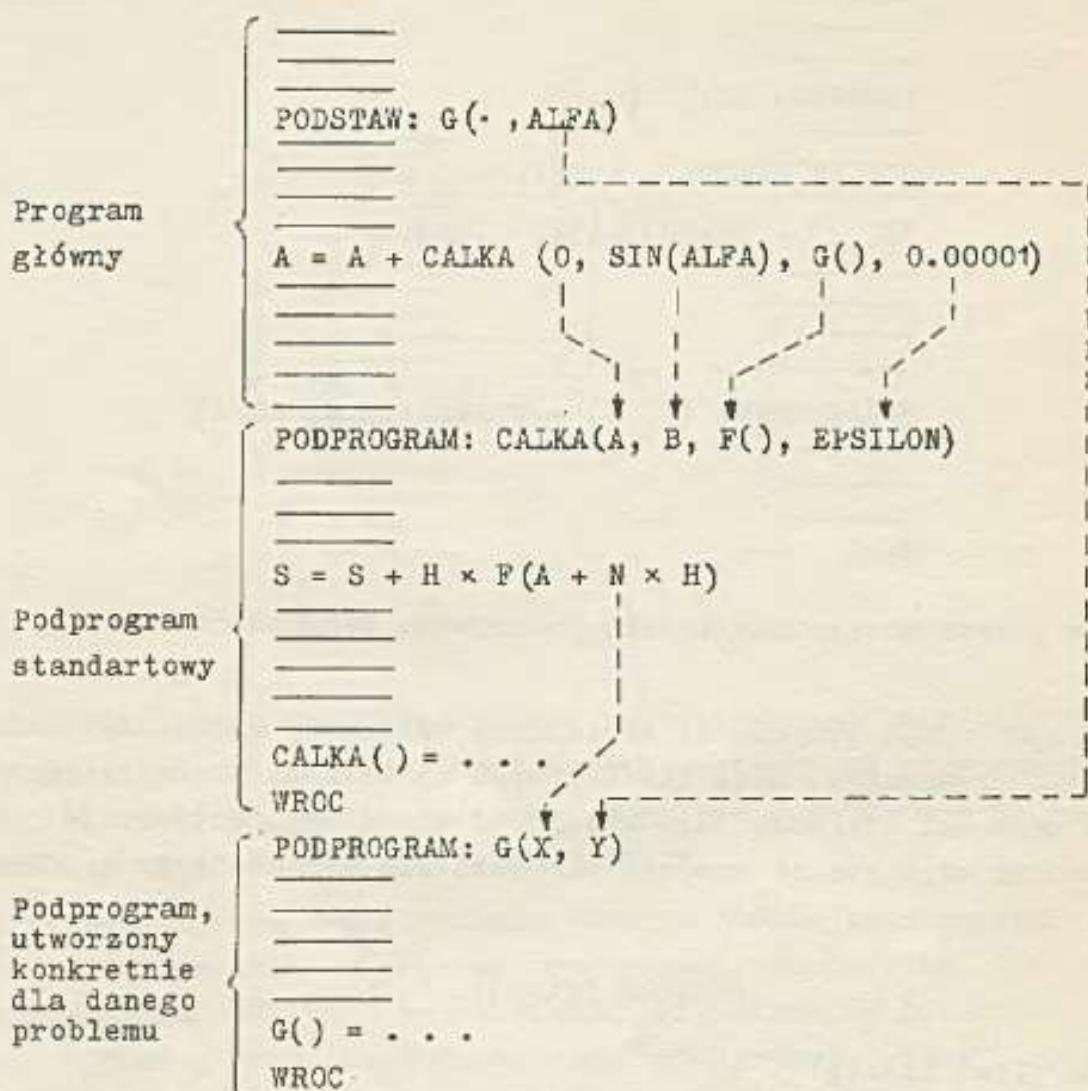
PODSTAW: TRANS(A)

Jeśli wszystkie argumenty podprogramu zostały podstawione przy pomocy rozkazów PODSTAW, wówczas formula operacyjna, wywołująca podprogram, może mieć postać:

(U, V) = TRANS

Możliwość ta posiada duże znaczenie dla korzystania z podprogramów II rzędu, przystosowanych do działania na podprogramach o ustalonej ilości argumentów, gdy chcemy uwzględnić jeszcze pewną dodatkową ilość parametrów.

Pokazuje to następujący przykład:



W powyższym przykładzie CALKA jest funkcjonakiem, G() jest funkcją. Podprogram CALKA jest podprogramem II rzędu, podprogram G() jest podprogramem I rzędu. Argumentami podprogramu CALKA są:

granice całkowania A i B,
funkcja podcałkowa F(),
dokładność obliczania całki EPSILON.

Podprogram G() zależy od dwóch argumentów: zmiennej X i parametru Y.

Program główny przy pomocy rozkazu PODSTAW określa parametr Y funkcji G jako równy ALFA. Następnie poprzez formułę arytmetyczną wywołuje podprogram CALKA, określając w nim granice całkowania,

funkcję podcałkową oraz żądaną dokładność. /Funkcją podcałkową jest tutaj funkcja G(), nie zaś SIN(ALFA)/. Dla uniknięcia nieporozumień piszemy zawsze

F()

gdy chodzi o funkcję, zaś

F(lista argumentów)

gdy chodzi o wartość funkcji.

Należy zauważyć, że jeden argument funkcji G pozostał niero- kreślony - jest nim zmienna całkowania.

W podprogramie CALKA wykorzystujemy podprogram G wywołując go formułą arytmetyczną i przekazując mu pierwszy /dotychczas nie podstawiony/ argument jako równy wartości wyrażenia arytmetycznego $A + N \times H$. W momencie wywołania podprogramu G wszystkie jego argumenty są określone.

Przykład powyższy objaśnia możliwość zmiany ilości parametrów funkcji podcałkowej przez program główny przy wykorzystaniu tylko jednego podprogramu standartowego CALKA. Z przykładu tego widać również, że funkcją podcałkową może być funkcja o dowolnej liczbie argumentów, mimo wykorzystywania tej funkcji przez podprogram również, że funkcją podcałkową może być funkcja o dowolnej liczbie argumentów, mimo wykorzystywania tej funkcji przez podprogram CALKA jako funkcji jednej zmiennej. Jest to możliwe z jednej stro- ny dzięki rozkazowi PODSTAW, z drugiej - dzięki możliwości nie wpisania kropki po ostatnim podstawianym argumencie.

System SAKO dopuszcza uogólnienie zasady, zilustrowanej powyższym przykładem, na przypadek podprogramów dowolnie wysokiego rzędu, o dowolnej ilości argumentów, które są również podprogra- mami dowolnie wysokiego rzędu o dowolnej ilości parametrów.

SKOROWIDZ POJĘĆ PODSTAWOWYCH

- Arytmometr 12
autokod 4
- Binarny zapis liczb 15
blok liczbowy 29
- Czas dostępu średni 11
czytnik 10
- Dalekopis 8
dane 6
deklaracje 22
dziurkarka 10
- Elementy bloku 29
- Formuła arytmetyczna 63
- bulowska 64
- operacyjna 64
funkcje języka 34, 35
- definiowane 84
- Identyczność 18
iloozyn bulowski 19
indeks 29
informacja wejściowa 6
- wyjściowa 6
- Język maszyny 3
- Kolejność wykonania 23
komentarze 23
- Liczba całkowita 14, 17, 28
- ułamkowa 13, 15, 26
lista 41
- Moc działań 37, 38
- Nadmiar 27, 39
następny rozkaz 23
nawiasy 24
negacja 19
numery 34
- Oktalowy zapis słów bulowych 21
- Pamięć 11
- wewnętrzna 12
pojemność pamięci 11
powtarzania przerwanie 56
- wykonanie 57
program 3
- kodujący 4
- w języku autokodu 4
przesunięcie cykliczne 20
- Rozdział 12, 23, 42
rozkazy 22
- SAKO 3
separatory 24
skala binarna 13, 15, 27
- dziesiętna 13, 15, 27
- obliczenia 39
- słowo 15
- binarne 15
- bulowskie 17, 29
- długie 11
- krótkie 11
- spacja 24
sterowanie 13
stolik operatora 13
- suma bulowska 18

- Taśma pięciokanałowa 8
tekst 24
- Urządzenia wejściowe 8
- wyjściowe 10
- Wskaźnik nadmiaru 28, 39
wymiary bloku 29
wyniki 6
wyrażenia 24
- arytmetyczne 36
- wyrażenia bulowskie 39
- całkowite 38
- ułamkowe 38
- Zakres indeksu 29
zmienna indeksowana 33
- prosta 31, 32
znaki dalekopisowe 9, 23, 24
- działań i relacji 24
- liczby 70*)
zwroty 24

L I S T A Z D A N J J E Z Y K A S A K O

1. DEKLARACJE

- ✓ 1. ROZDZIAŁ : nazwa
- ✓ 2. PODPROGRAM : (Lista wyników) *
 * nazwa (Lista argumentów)
- ✓ 3. BLOK (n, m, ..., k) : lista
- ✓ 4. STRUKTURA (I, J, ..., K) : lista
- ✓ 5. TABLICA (n, m, ..., k) : nazwa
- ✓ 6. TABLICA ORTALNA (n, m, ..., k) : nazwa
- ✓ 7. CAŁKOWITE : lista
- ✓ 8. SKALA DZIESIĘTNIA PARAMETROW n
- ✓ 9. SKALA BINARNA PARAMETROW n
- ✓ 10. JĘZYK SAS
- ✓ 11. JĘZYK SAKO
- ✓ 12. KONIEC

3. ROZKAZY ARYTMETYCZNE I BULOWSKIE

- ✓ 1. A = B
- ✓ 2. A = B
- ✓ 3. (Lista wyników) = nazwa podprogramu (Lista argumentów)
- ✓ 4. PODSTAW : nazwa podprogramu (Lista)
- ✓ 5. USTAW SKALE DZIESIĘTNIE : I
- ✓ 6. USTAW SKALE BINARNE : I
- ✓ 7. ZWIĘKSZ SKALE DZIESIĘT-
- NIE o I : lista
- ✓ 8. ZWIĘKSZ SKALE BINAR-
- NIE o I : lista

4. ROZKAZY WEJŚCIA I WYJŚCIA

- ✓ 1. CZYTAJ : lista
- ✓ 2. CZYTAJ OTALNIĘ : lista
- ✓ 3. CZYTAJ WIERZĄ : nazwa bloku
- ✓ 4. DRUKUJ (I, J) : lista zmien-nych
- ✓ 5. DRUKUJ WIERZĄ : nazwa bloku
- ✓ 6. DEKLUJ SŁOWO : lista zmien-nych
- ✓ 7. TEKST :
- V 8. TEKST WIERZĄ n :
- V 9. SPACJA n
- ✓ 10. LDRIA n

5. ROZKAZY WSPÓŁPRACY Z BBNSM

- ✓ 1. CZYTAJ Z BBNSA OD I : lista
- ✓ 2. PISS NA BBNSB OD I : lista