



BIBLIOTEKA

— WAT —

Dział

Nr inw.

59522

Sygn.

II-37149

INSTYTUT MATEMATYCZNY POLSKIEJ AKADEMII NAUK

LEON ŁUKASZEWCZ, ANTONI MAZURKIEWICZ

SYSTEM AUTOMATYCZNEGO KODOWANIA
SAKO

WYDANIE DRUGIE
POSZERZONE



1966

WROCŁAW-WARSZAWA-KRAKÓW

ZAKŁAD NARODOWY IMIERIA OSSOLIŃSKICH
WYDAWNICTWO POLSKIEJ AKADEMII NAUK

Dingański

Niniejsza praca została opublikowana po raz pierwszy w 1962 r. przez Biuro Kształcenia i Doskonalenia Kadra Naukowych PAN dla potrzeb uczestników XVII kursu zastosowań matematyki.

Zakład Narodowy im. Ossolińskich Wydawnictwo Polskiej Akademii Nauk. Wrocław. 1966. Wyd. II. Nakład: 2000+100 egz. Objętość: 10,85 ark. wyd.; 13,75 ark. druk. Papier ofsetowy III kl., 80-gramowy. A1. Powielono w Warszawskiej Drukarni Naukowej Warszawa, ul. Śniadeckich 8 Zara. 292/0 E-79. Cena: 25,- zł

110/66

SPIS TREŚCI

OD AUTORÓW	5
WSTĘP	7
Rozdział 1 WSTĘPNE WIADOMOŚCI O MASZYNACH CYFROWYCH I PROGRAMOWANIU	11
1. Schemat przedstawiania informacji	11
2. Części składowe maszyn cyfrowych XYZ i ZAM-2	12
3. Zapis dziesiętny liczb w maszynie	18
4. Zapis binarny liczb w maszynie	19
5. Ogólna struktura programów w języku SAKO ..	26
6. Metodyka przygotowania programu	29
Rozdział 2 PRZYKŁADY WPROWADZAJĄCE	31
1. Obliczanie większego pierwiastka równania kwadratowego przy A dodatnim	31
2. Obliczanie większego pierwiastka równania kwadratowego przy A dodatnim lub ujemnym ..	35
3. Tablicowanie wartości wielomianu	38
4. Mnożenie macierzy kwadratowej przez macierz względem niej przedstawioną	42
5. Obliczanie pierwiastka sumy kwadratów w zmiennej skali	54
6. Cechowanie współczynników macierzy	58
7. Analiza wiersza tytułowego	64
8. Obliczanie wartości funkcji	66
9. Tablicowanie pierwiastków funkcji przestępnej	70
10. Tablicowanie funkcji $SI(X)$	76
11. Rozwiązywanie równania różniczkowego	84
Rozdział 3 FORMA I OPIS PODSTAWOWYCH POJĘĆ W JĘZYKU SAKO	94
1. Liczby	94
2. Zmienne	99
3. Numery	101
4. Funkcje	102
5. Wyrażenia arytmetyczne	104
6. Wyrażenia bulowskie	108
7. Lista	110

Rozdział 4 FORMA I OPIS POSZCZEGÓLNYCH ROZKAZÓW SAKO	112
1. Deklaracja	112
2. Rozkazy sterujące	123
3. Rozkazy arytmetyczne i bulowskie	136
4. Rozkazy wejścia i wyjścia	142
5. Rozkazy współpracy z bębnem	151
6. Komentarze	153
Rozdział 5 PROGRAMY W JĘZYKU SAKO	155
1. Terminologia i oznaczenia	156
2. Ogólne właściwości systemu korzystania z podprogramów w języku SAKO	160
3. Budowa programów SAKO	161
4. Korzystanie z podprogramów napisanych w języku SAKO	164
Rozdział 6 SIECI DZIAŁAŃ	171
Rozdział 7 STRUKTURA SYNTAKTYCZNA JĘZYKA SAKO	181
1. Symbolika opisu	181
2. Opis wyrażeń SAKO	
Rozdział 8 SYSTEM OPERACYJNY	199
1. Przygotowanie taśmy z programem	199
2. Znaczenie dyrektywy L	201
3. Wprowadzenie programu SAKO do maszyny	202
4. Uruchomienie programu	205

OD AUTORÓW

System Automatycznego Kodowania SAKO stworzony został w celu zasadniczego usprawnienia kodowania programów dla maszyn XYZ oraz ZAM - 2. Programy te, napisane w łatwym do opanowania języku SAKO, są następnie sprowadzane do postaci ostatecznej przez odpowiedni program tłumaczący, zapisany na stałe w pamięci bieżącej maszyny. W ten sposób sama maszyna wykonuje praceochłonne złożoności kodowania, które w klasycznym programowaniu obciążają człowieka, dzięki czemu łączny czas przygotowania programu może być wielokrotnie skrócony. Sprawność programu wynikowego, uzyskanego przez program tłumaczący SAKO jest na ogół taka, jak gdyby program ten wykonał sprawny programista.

Niniejszy podręcznik ma na celu stopniowe wprowadzenie czytelnika w programowanie w języku SAKO. Dla zrozumienia podanych w podręczniku przykładów wystarczająca jest znajomość jedynie podstaw matematyki wyższej, jaką z zasady dysponuje każdy pracownik w wyższym wykształceniem, posługujący się matematyką jako narzędziem pracy.

Rozdział 1 zawiera wiadomości wstępne o programowaniu, a szczególnie niezbędne dla programisty wiadomości o maszynie matematycznej ZAM - 2. W ten sposób nie zakłada się u czytelnika żadnej uprzedniej znajomości maszyn matematycznych, aczkolwiek wiedza taka byłaby niewątpliwie pewnym ułatwieniem dla zrozumienia tego rozdziału.

W rozdziale 2 podano 11 przykładów pełnych programów w ję-

zyku SAKO, ułożonych zgodnie z metodą podaną na końcu rozdziału wstępnego. Przykłady te należy przerabiać kolejno, na końcu każdego z nich umieszczony jest wykaz stron rozdziału 3, 4 i 5, które powinny być przerobione w związku z omawianym przykładem. Rozdział 6 podaje zbiór reguł, związanych z budową sieci działań. Rozdział 7 podaje strukturę syntaktyczną języka SAKO, to jest sformalizowany opis formy dowolnych wyrażeń i zwrotów języka SAKO. Rozdział 8 poświęcony jest sposobom wprowadzania i uruchamiania programów.

Przy pierwszym czytaniu podręcznika można opuścić § 4 rozdziału 1, punkt a w § 1 rozdziału 2 oraz przykłady § 6 i § 7 rozdziału 3, związane z binarnym przedstawieniem liczb w maszynie oraz działaniami bulowskimi.

System Automatycznego Kodowania SAKO opracowany został w latach 1959 - 1960 przez zespół pracowników naukowych Zakładu Aparatów Matematycznych PAN w składzie: Leon Łukaszewicz, Antoni Mazurkiewicz, Jan Borowiec, Jowita Koncewicz, Maria Łacka, Stefan Sawicki, Jerzy Swianiewicz, Piotr Szorc, Alfred Szurman i Andrzej Wiśniewski.

Ponadto w pracach nad SAKO udział wzięli: Ludwik Czaja, Danuta Kosecka, Zdzisław Wrotek, Jacek Witaszek, Ewa Zaborowska i Jacek Moszczyński.

Rozdział 8 podręcznika został w całości opracowany i napisany przez Jerzego Swianiewicza.

Wszystkie przykłady, podane w podręczniku, zostały sprawdzone na maszynie ZAM - 2 przez Andrzeja M. Wiśniewskiego, za co autorzy składają mu serdeczne podziękowanie.

W S T E P

System Automatycznego Kodowania SAKO opracowany został w celu znacznego zmniejszenia wysiłku i czasu, potrzebnych dla przygotowania programów. Jego zasadniczą rolą jest pełnienie funkcji tłumacza pomiędzy programistą a maszyną. Programista zapisuje program w Języku podobnym do języka przyjętego ogólnie w matematyce, a program tłumaczący /translator/ SAKO przekształca go na program zapisany w języku maszyny. Program ten stanowi bezpośredni, 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 języku maszyny nazywamy taki jego zapis, w którym wyszczególniony jest każdy rozkaz maszyny składający się na ten program. Przykładem takiego języka jest na przykład 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,

- znakoma przejrzystość zakodowanego programu,

- niemal nieuniknione, liczne błędy w programie,

- konieczność bezproduktywnej pracy maszyny przy uruchamianiu programu,

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

System Automatycznego Kodowania SAKO, podobnie jak inne autokody, pozwala na przewyścielenie tych trudności. W systemie takim programista zapisuje metodę rozwiązania problemu przez maszynę w specjalnym języku sformalizowanym, zbliżonym do zapisu stosowanego powszechnie w matematyce. Nauczenie się i posługiwanie tym językiem nie przedstawia specjalnych trudności. Na podstawie zapisu w SAKO maszyna sama tłumaczy program na język 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 y n i k c w y, zapisany w języku maszyny. Dokonuje tego specjalny program tłumaczący /translator/ zapisany na stałe w pamięci bębnowej maszyny. W stosunku do programu tłumaczącego program zapisany w autokodzie stanowi dane wejściowe, natomiast wynikiem jest ten sam program, zapisany w języku maszyny.

Program tłumaczący dla systemu SAKO składa się z około 10 000 rozkazów w języku maszyny. Jego obszerność i złożoność spowodowana jest faktem, że musi on całkowicie zastąpić doświadczonego i sprawnego programistę.

Autokod pozwala na szerokie stosowanie maszyn cyfrowych do konkretnych problemów obliczeniowych. Czas bowiem potrzebny na wyszkolenie personelu programującego maszynę w systemie SAKO jest wielokrotnie krótszy od czasu potrzebnego na wyszkolenie programistów, programujących w języku maszyny. Pozwala to specjalistom z różnych dziedzin na przyswojenie sobie zasad pracy w autokodzie w ciągu niewielu dni i w 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 ZWM - 2.

Rozdział 1

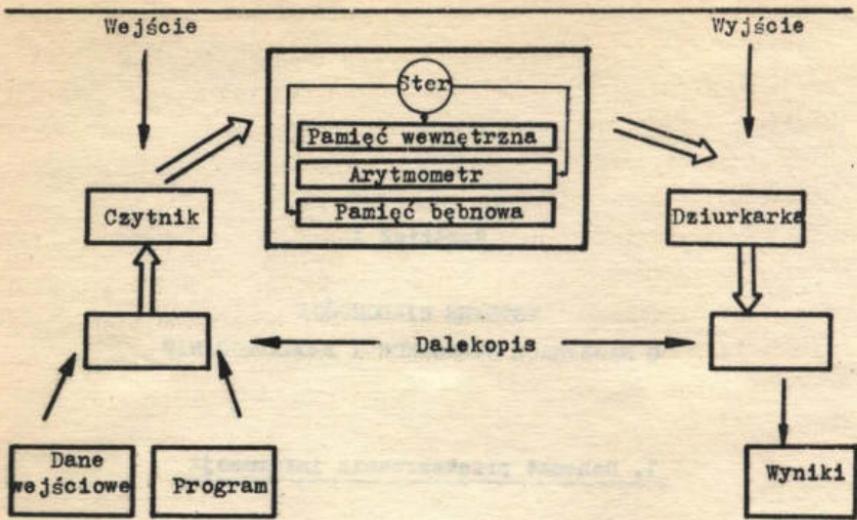
WSTĘPNE WIADOMOŚCI O MASZYNACH CYFROWYCH I PROGRAMOWANIU

1. Schemat przetwarzania informacji

Automatyczne maszyny cyfrowe służą, w najogólniejszym ujęciu, do przetwarzania informacji wejściowych, zwanych danymi i - na wyjściowe, zwane wynikami. 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 stanowić 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 na przykład 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 możliwości ułożenia takiego programu w całej rozciągłości, można zaznaczyć, że przy najmniej teoretycznie programy tego typu mogą być zapisane w języku SAKO.



Rysunek 1
Schemat przetwarzania informacji
przez maszynę cyfrową

Obrazowy schemat przetwarzania informacji przez maszynę przedstawiony jest na rysunku 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.

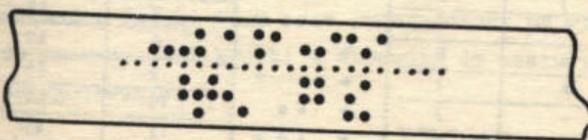
2. Części składowe maszyn cyfrowych XYZ i ZAM-2

a. 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 kombinacji. Odcinek taśmy z wydziurkowaną informacją "22 LIPCA 1961" przedstawiony jest na rysunku 2.

Dziurkowanie taśmy odbywa się przy pomocy dalekopisu. Jest urządzenie zbliżone do zwykłej maszyny do pisania, z tą różnicą, że równocześnie z naciśnięciem każdego klawisza następuje, obok napisania litery na arkuszu, wydziurkowanie na taśmie odpowiedniej kombinacji dziurek.



Rysunek 2

Odcinek taśmy
z wydziurkowaną informacją "22 LIPCA 1961"

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 na ogół 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 za pomocą dalekopisu posiada tę cenną zaletę, że wraz z taśmą otrzymujemy pisany dokument, będący dokładnym odpowiednikiem jej treści. Przy jego pomocy możemy na przykład wykryć niemal wszystkie błędy powstałe przy dziurkowaniu taśmy. Na tej podstawie, posługując się otrzymaną taśmą

Tablica 1

Znaki dalekopisowe dla wejścia i wyjścia
na taśmie dziurkowanej

Cyfry	Litery	Taśma	Wartość cyfr	Wartość liter
		5 4 3 2 1		
C y f r y				
1	A	•	0	32
2	B	• •	1	33
*	C	• • •	2	34
4	D	• •	3	35
(E	• • •	4	36
)	F	• • •	5	37
7	G	• • • •	6	38
8	H	• •	7	39
=	I	• • •	8	40
=	J	• • •	9	41
-	K	• • • •	10	42
V	L	• • •	11	43
LINIA	M	• • • •	12	44
SPACJA	N	• • • •	13	45
*	O	• • • • •	14	46
0	P	•	15	47
>	Q	• • •	16	48
:	R	• • •	17	49
3	S	• • • •	18	50
→	T	• • •	19	51
5	U	• • • •	20	52
6	V	• • • •	21	53
/	W	• • • • •	22	54
x	X	• • •	23	55
9	Y	• • • •	24	56
+	Z	• • • • •	25	57
L i t e r y				
.	.	• • •	26	58
n	?	• • • •	27	59
P.K.	æ	• • • • •	28	60
*BLAD/	*BLAD/	• • • • • •	29	61
			30	62
			31	63

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 za pomocą dalekopisów znajdują się w dalszej części podręcznika w postaci programów, danych wejściowych i wyników.

Informacje, wydrukowane 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 krótki w stosunku do czasu niezbędnego na ich przygotowanie.

b. Urządzenia wyjściowe. 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 formie arkuszowym. Oczywiście, tym samym kombinacjom dziurek w rzadkach taśmy odpowiadają te same znaki dalekopisowe, co w urządzeniach wejściowych. W ten sposób np. jeden i ten sam dalekopis może być używany do dziurkowania taśmy i drukowania wyników. Ponadto, wyniki wydziurkowane 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.

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

Pamięć składa się z pewnej ilości podstawowych komórek, których zawartość informacyjną nazywamy słowem długim. Słowo takie może przedstawić sobą liczbę ułamkową lub słowo bulowskie /patrz następne rozdziały/. Każda połówka słowa długiego nosi nazwę słowa krótkiego; słowo krótkie może przedstawiać rozkaz będący składnikiem programu lub liczbą 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 w niej 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.

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 wewnętrznej umieszczane są kolejno w odpowiednim porządku.

W pamięci bębornej umieszczamy materiał liczbowy oraz te rozdziały programu, które nie są aktualnie opracowywane, a które są używane we wcześniejszych lub późniejszych etapach programu. Przykładem informacji z pamięci bębornej do wewnętrznej lub odwrotnie, kierowane jest przez program, znajdujący się w pamięci wewnętrznej. Wobec nadającego charakteru pamięci wewnętrznej w stosunku do bębornej, 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 wewnętrzna uzupełniamy w XYZ i ZAM-2 powolną, lecz za to bardzo pojemną pamięcią bębnową.

d. A r y t m o m e t r . 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, za czas pomnożenia lub podzielenia - 0,003 sekundy. Do czasów tych dodaje się jednak czasy oczekiwania na pobranie rozkazów i liczb z pamięci wewnętrznej.

e. S t e r o w a n i e . Sterowanie jest urządzeniem koordynującym pracę pozostałych zespołów maszyny. Dodatkowo maszyna może być sterowana ręcznie za pośrednictwem przycisków i klawiszy znajdujących się na stoliku operatora.



3. Zapis dziesiętny liczb w maszynie

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

a. **Liczby ułamkowe.** Dla zapисania liczby ułamkowej w pamięci maszyny potrzebna jest jedna komórka pamięci. Wyobrazić sobie możemy, że komórka ta składa się z 12 pozycji:

0	1	2	3	4	5	6	7	8	9	10	

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 skali dziesiętnej liczby, która sygnalizowana jest maszynie rozkazem USTAW SKALE DZIESIĘTNIE i deklaracją SKALA DZIESIĘTNA PARAMETRÓW.

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

b. L i c z b y c a ɿ 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 dziesiętne 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

4. Zapis binarny liczb w maszynie¹

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.

¹ Paragraf ten można opuścić przy pierwszym czytaniu podręcznika.

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	0	1	0	1	1	0	1	1	0	1	1	1	0	1	1	0	1	1	0	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.

a. **Liczby całkowite.** Pozycja zerowa słowa określa znak liczby:

0 - 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.

1	000011010	01100	0
znak	część całkowita	część ułamkowa	

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 spo-

sób wykonywania działań na tych liczbach. Skala binarna jest sygnalizowana maszynie rozkazem USTAW SKALE BINARNIE oraz deklaracją SKALA BINARNA PARAMETRÓW, na przykład:

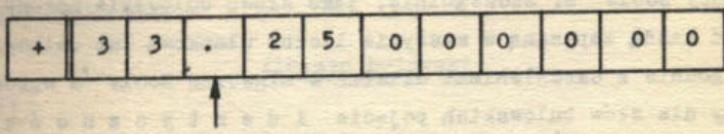
USTAW SKALE BINARNIE : 12

SKALA BINARNA PARAMETRÓW : 8

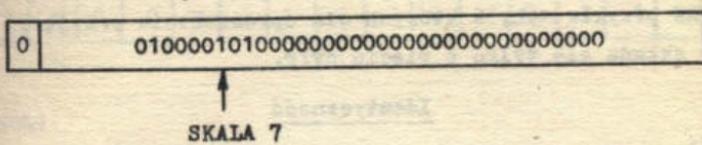
Każdej skali dziesiętnej odpowiada skala binarna według następującej tabelki:

SKALA DZIESIĘTNA	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:



b. L i c z b y c a łącz k o w i t e . Na zapisanie liczby całkowitej w maszynie wykorzystujemy tylko połowę słowa, to znaczy 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 całkowita, 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 liczbę 39 maszyna traktuje jako następujące słowo:

0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	16	17	18	19	20	21	22	23	34	35				

c. Słowo i działania bulowskie. Przez słowo bulowskie rozumiemy słowo, którego kolejne cyfry 0 lub 1 traktujemy jako elementy dwuelementowej algebry Boole 'a. Szczególnie, jako słowo bulowskie możemy traktować każdą zapisaną w maszynie liczbę ułamkową lub całkowitą.

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 i cyklicznego słowa. Sens tych działań objaśniony jest na przykładach, w których dla uproszczenia przyjęto, że słowo składa się tylko z pięciu cyfr.

Identyczność

Zapis:

$$A = B$$

Przykład:

0	1	0	1	1
---	---	---	---	---

A

\equiv

0	1	0	1	1
---	---	---	---	---

B

Znaczenie:

Dwa słowa są identyczne, jeśli wszystkie cyfry jednego sło-

wa są identyczne z odpowiadającymi im cyframi drugiego słowa.

Suma bulowska

Zapis:

$$A \equiv B + C$$

Przykład:

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

Znaczenie:

Wynikiem sumy bulowskiej 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 w jednym lub drugim składniku sumy.

Iloczyn bulowski

Zapis:

$$A \equiv B \times C$$

Przykład:

$$\begin{array}{c} \boxed{0 \ 1 \ 0 \ 0 \ 0} \\ A \end{array} \equiv \begin{array}{c} \boxed{1 \ 1 \ 0 \ 1 \ 0} \\ B \end{array} \times \begin{array}{c} \boxed{0 \ 1 \ 1 \ 0 \ 1} \\ C \end{array}$$

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.

Negacja

Zapis:

$$A \equiv \neg B$$

Przykład:

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

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.

Przesunięcie cykliczne w lewo

Zapis:

$$A = B * n$$

n - liczba całkowita dodatnia

Przykład:

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

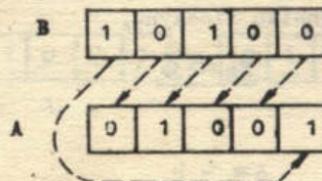
A B n

Znaczenie:

Wynikiem przesunięcia słowa B w lewo o jeden:

$$A = B * 1$$

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 słowie B na zerowej 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.

Przesunięcie cykliczne w prawo

Zapis:

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

Przykład:

$$\begin{array}{|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 1 \\ \hline \end{array} \equiv \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 1 & 0 \\ \hline \end{array} * (-3)$$

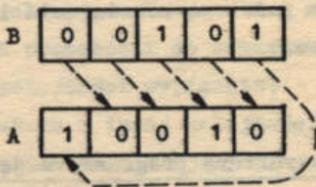
A B n

Znaczenie:

Wynikiem przesunięcia słowa B w prawo o jeden:

$$A = B * (-1)$$

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.

d. 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, na przykład:

130.267.304.256

Nazwa "system oktalowy" jest równoważna nazwie "system ósemkowy". W systemie tym podstawą rozwinięcia liczb jest nie liczba dziesięć lecz osiem.

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 reguł. 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 swoją 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 wypisywania rozkazów przez programistę, od kolejności wykonania tych rozkazów przez maszynę. Na ogół 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

+ - x / * - = >

4. Separatory i nawiasy

. , : ()

5. Spacje /odstępy między znakami/.

Pewne określone ciągi znaków /nie licząc spacji/ nazywać będziemy zwrotami języka SAKO. Przykładami zwrotów są:

CZYTAJ :

NASTĘPNY

GDY

Pewne ciągi znaków o określonej budowie nazywać będziemy wyrażeniami. Przykładami wyrażeń są:

liczby

zmienne

funkcje

listy zmiennych

wyrażenia arytmetyczne

wyrażenia bulowskie

Pewne ciągi dowolnych znaków o jednoznacznie określonym znaku początkowym i końcowym nazywać będziemy tekstaniami:

X1 = PIERWIASTEK =

Zdania SAKO zbudowane są jako określone ciągi zwrotów, wyrażeń i tekstów. Budowa zwrotów, wyrażeń i tekstów, kolejność ich występowania musi być zgodna z regułami budowy zdania SAKO /patrz rozdz. 7 podręcznika/.

Każde zdanie SAKO rozpoczyna się od nowego wiersza. Przez wiersz rozumiemy odcinek taśmy zawarty między dwoma kolejnymi znakami PK. Ilość wierszy zajętych przez zdanie wynika wyłącznie z jego budowy. Na ogół zdania SAKO zamykają się w jednym wierszu.

6. Metodyka przygotowania programu

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

1. pełne sformułowanie problemu,
2. metoda rozwiązania uwzględniająca specyfikę pracy maszyn cyfrowych,
3. dyskusja skali 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 stanoi tylko jedną z wielu pozycji, jakie powinien wykonać programista.

W tym momencie należy zatrudnić się w zakresie dokumentacji, w tym ostatecznej dokumentacji rozwiązania problemu.

W tym momencie należy zatrudnić się w zakresie dokumentacji, w tym ostatecznej dokumentacji rozwiązania problemu.

W tym momencie należy zatrudnić się w zakresie dokumentacji, w tym ostatecznej dokumentacji rozwiązania problemu.

W tym momencie należy zatrudnić się w zakresie dokumentacji, w tym ostatecznej dokumentacji rozwiązania problemu.

W tym momencie należy zatrudnić się w zakresie dokumentacji, w tym ostatecznej dokumentacji rozwiązania problemu.

W tym momencie należy zatrudnić się w zakresie dokumentacji, w tym ostatecznej dokumentacji rozwiązania problemu.

W tym momencie należy zatrudnić się w zakresie dokumentacji, w tym ostatecznej dokumentacji rozwiązania problemu.

W tym momencie należy zatrudnić się w zakresie dokumentacji, w tym ostatecznej dokumentacji rozwiązania problemu.

W tym momencie należy zatrudnić się w zakresie dokumentacji, w tym ostatecznej dokumentacji rozwiązania problemu.

Rozdział 2

PRZYKŁADY WPROWADZAJĄCE

1. Obliczanie większego pierwiastka równania kwadratowego przy A dodatnim

Problem:

Należy ułożyć program obliczającywiększy pierwiastek równania

$$A \cdot X^2 + B \cdot X + C = 0,$$

przy założeniuach:

$$B^2 - 4AC > 0,$$

$$1 < A < 10,$$

$$|B|, |C| < 10.$$

Metoda rozwiązania:

Zastosujemy wzór

$$X_2 = (-B + \sqrt{B^2 - 4AC}) / 2A$$

Skala:

Latwo sprawdzić, że wszystkie wartości pośrednie i końcowe, wynikające z zastosowanego wzoru nie przekraczają $10^3 = 1000$, wystarczy więc przyjąć skalę 3.

Dokładność wyników:

Załóżmy, że interesują nas tylko trzy cyfry znaczace po kropce. Ponieważ maszyna, licząc w skali 3, operuje siedmioma znakami po kropce, łatwo uzyskuje się żądaną dokładność.

Postać danych wejściowych:

Współczynniki A, B, C, wypisujemy kolejno pod sobą na formularzu danych wejściowych, np.

$$\begin{array}{r} 3 \cdot 784 \\ 7 \cdot 345 \\ -3 \cdot 901 \end{array}$$

Jednocześnie z wypisywaniem tych danych otrzymujemy odpowiednie wydziurkowany odcinek taśmy.

Postać wyników:

Wynik chcemy uzyskać w postaci liczby, zaokrąglonej do trzech znaków po kropce i wypisanej w oddzielnym wierszu formularza. Na zapisanie tej liczby rezerwujemy trzy pozycje przed i trzy po kropce:

± ddd.ddd

gdzie d oznacza cyfrę dziesiątną.

Sieć działań:

Z powodu prostoty przykładu sieć działań może być opuszczona.

Objaśnienie sieci działań:

Program:

Program w języku SAKO może być napisany następująco:

```

K)   1)   OBLCZANIE WIEKSZEGO PIERWIASTKA PRZY A DODATNIM
      USTAW SKALE DZIESIETNIE:3
      CZYTAJ: A,B,C
      X2=(-B+PUK(B*2-4*A*C))/(2*A)
      LHLIA
      DRUKUJ(3,3):X2
      STOP
      KONIEC
  
```

Objaśnienie programu:

Powyższy program składa się z ciągu zdań w języku SAKO, z

2. Dane wejściowe, wyniki i programy podane są w postaci, jaką utrzymuje się na dalekopisie przy dziurkowaniu taśmy.

których każde zajmuje oddzielny wiersz. W programie tym kolejność wykonywania zdań programu jest zgodna z kolejnością ich wypisania.

Pierwszym zdaniem jest KOMENTARZ, co jest zaznaczone przez umieszczenie litery K i nawiasu zamykającego na początku wiersza. Komentarze nie mają żadnego znaczenia dla wykonania programu, a służą dla zwiększenia przejrzystości jego zapisu lub - jak w tym przypadku - dla odróżnienia go od innych programów.

Drugie zdanie określa skalę, jaką mają wszystkie zmienne układowe występujące w programie.

Rozkaz CZYTAJ - powoduje wczytanie do maszyny trzech kolejnych liczb znajdujących się na wejściu i oznaczenie ich symbolami A, B, C.

Zakładamy, że taśma z wydziurkowanymi danymi została dołączona do wejścia maszyny przed rozpoczęciem wykonywania programu.

Kolejnym zdaniem jest FORMUŁA ARYTMETYCZNA, która powoduje obliczenie wielkości X2. W podanej formule symbol $\sqrt{}$ został zastąpiony symbolem PWK - pierwiastek kwadratowy /por. tab. 2/. Istotne w formule jest zaznaczenie operacji mnożenia przez odpowiadający jej krzyżyk x.

Zdanie LINIA powoduje rozpoczęcie drukowania wyników w nowej linii.

Zdanie DRUKUJ powoduje wydrukowanie /przez wydziurkowanie taśmy dalekopisowej/ wielkości X2, rezerwując znak oraz 3 pozycje przed kropką i zaokrąglając wynik do 3 cyfr po kropce. Razem, na wydrukowanie X2 zarezerwowanych zostało $1+3+1+3=8$ pozycji arkusza wydawniczego /poza cyframi liczby jedno miejsce na znak liczby i jedno na kropkę/.

Zdanie STOP 1 powoduje zatrzymanie się maszyny. Po ewentualnym dołączeniu nowych danych wejściowych i ponownym naciśnięciu przycisku "START" wykonanie programu zostanie powtórzone od rozkazu w numerze 1.

Zdanie KONIEC sygnalizuje koniec programu.

Sprawdzenie programu:

Sprawdzenia poprawności programu można dokonać, przyjmując pewne wielkości A, B, C, dla których wynik X2 jest znany, a następnie wykonując program przy tych wielkościach i sprawdzając X2. Na przykład przy

$$A = 1.0$$

$$B = 5.0$$

$$C = 4.0$$

powinniśmy otrzymać $X2 = -1.000$.

Sprawdzenie wyników:

Najprościej przeprowadza się sprawdzenie otrzymanego wyniku X2 porównując go do równania wyjściowego i sprawdzając czy równanie te jest spełnione.

W przypadku programu dającego dużą ilość wyników sprawdzamy zazwyczaj tylko niektóre z nich.

Dla wartości współczynników podanych przykładowo w punkcie "Pastać danych wejściowych" wynik obliczeń wynosi +0.434

Nowe pojęcia SAKO użyte w przykładzie:

Dla dokładniejszej analizy tego przykładu należy przejrzeć następujące pozycje rozdziału 1, 3, 4:

Zapis dziesiętny w maszynie liczb	s. 18
Liczba ułamkowa	s. 94
Liczba całkowita	s. 96
Zmienna prosta	s. 99
Wyrażenia arytmetyczne	s. 104
Lista	s. 110
KONIEC	s. 122
STOP	s. 135
Formuła arytmetyczna	s. 136
USTAW SKALE	s. 140
CZYTAJ	s. 142
DRUKUJ	s. 145
LINIA	s. 150
Komentarz	s. 153

2. Obliczenie większego pierwiastka
równania kwadratowego przy A dodatnim lub ujemnym

Problem:

Należy ułożyć program, obliczający większy pierwiastek równania kwadratowego:

$$Ax^2 + Bx + C = 0$$

przy założenях:

$$B^2 - 4AC > 0$$

$$1 < |A| < 10$$

$$|B|, |C| < 10$$

W odróżnieniu od przykładu 1, współczynnik A może być tutaj dodatni lub ujemny.

Metoda rozwiązania:

Zastosujemy jeden z następujących wzorów, w zależności od znaku A:

Gdy $A > 0$: $x_2 = \left(-B + \sqrt{B^2 - 4AC} \right) / 2A$ - wzór /1/

Gdy $A < 0$: $x_2 = \left(-B - \sqrt{B^2 - 4AC} \right) / 2A$ - wzór /2/

Skala - jak w przykładzie 1.

Dokładność wyników - jak w przykładzie 1.

Postać danych wejściowych - jak w przykładzie 1.

Postać wyników - jak w przykładzie 1.

Siec działan:

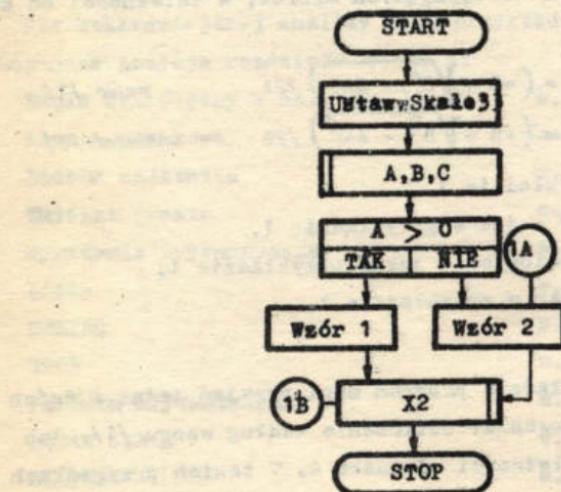
W niniejszym przykładzie program musi przyjąć jedną z dwóch możliwych dróg postępowania: obliczenie według wzoru /1/ lub według wzoru /2/ w zależności od znaku A. W takich przypadkach mówimy, że program posiada pewną strukturę logiczną. W przypadkach bardziej złożonej struktury logicznej, z zasadą posługujemy się graficznym środkiem pomocniczym, jukim jest sieć działań. Reguły, dotyczące pisania sieci działań, podane są w odrębnym rozdziale, s. 171, niniejszego opracowania. W rozpatrywanym przykładzie sieć działań jest przedstawiona na rysunku 3.

Objaśnienie sieci działań:

Początek sieci działań jest zawsze zaznaczony przez skrzynkę ovalną z umieszczonym nad nią napisem START.

Pierwsza skrzynka robocza oznacza ustawienie przyjętej skali, następna powoduje wczytanie do maszyny wielkości A, B, C. Na czynność wczytania wskazuje dodatkowa pionowa kreska przy lewym boku skrzynki.

Nastecną skrzynką jest skrzynka logiczna alternatywy. Wskazuje ona na jedną z dwóch możliwych do przyjęcia w dalszym ciągu dróg postępowania: jedną - w przypadku gdy warunek $A > 0$ jest spełniony /TAK/, drugą - w przypadku przeciwnym /NIE/. Każda z dwóch skrzynek znajdujących się na wyjściu skrzynki alternatywy symbolizuje odpowiednio zastosowanie wzoru /1/ albo zastosowanie wzoru /2/.



Rysunek 3

Sieć działań dla przykładu 2

Po skrzynkach powyższych następuje już tylko jedna skrzynka, symbolizująca wydrukowanie X2. Na czynność drukowania wskazuje dodatkowa pionowa kreska przy prawym boku tej skrzynki.

Ostatnia skrzynka w postaci ovalu z wpisany w niego słowem STOP, odpowiada zdaniu SAKO o tej samej nazwie.

Skrzynka z napisem WZÓR /2/ jest numerowana, gdyż najwyższej jednej skrzynki znajdującej się na wyjściu tej samej skrzynki logicznej może pozostać nienumerowana. Numerem tej skrzynki jest symbol 1A.

Skrzynka z napisem X2 jest numerowana, gdyż posiada więcej niż jedno wejście. Numerem jej jest symbol 1B.

Program:

Na podstawie podanej sieci działań łatwo już można napisać program:

```
K) OBLCZANIE WIEKSZEGO PIERWIASTKA PRZY A DOWOLNYM  
USTAW SKALE DZIESIETNIE:3  
  1) CZYTAJ: A,B,C  
      GDY A>0:NASTEPNY, INACZEJ 1A  
      WZOR(1):  
      X2=(-B+PKW(B*2-4*A*C))/(2*A)  
      SKOCZ DO 1B  
      WZOR(2):  
      1A) X2=(-B-PWK(B*2-4*A*C))/(2*A)  
      1B) LINIA  
          DRUKUJ (3..3):X2  
          STOP 1  
          KONIEC
```

Objaśnienie programu:

Skrzynce logicznej alternatywy, podanej w sieci działań na rysunku 3 odpowiada zdanie GDY programu w języku SAKO. Powoduje ono przejście do następnego zdania, jeżeli warunek $A > 0$ jest spełniony, lub przejście do zdania o numerze 1B w przypadku przeciwnym.

Zdanie SKOCZ DO 1B w programie powoduje bezpośrednie przejście do rozkazu, zazначенego numerem 1B.

Jak widać z porównania sieci działań i programu, każdemu numerowi, zaznaczonemu na sieci działań, odpowiada dokładnie jeden numer stojący przez odpowiednim zdaniem w programie.

Sprawdzenie programu - podobnie jak w przykładzie 1.

Sprawdzenie wyników - podobnie jak w przykładzie 1.

Nowe pojęcia SAKO, użyte w przykładzie:

Numery s. 101

SKOCZ s. 123

GDY s. 133

3. Tablicowanie wartości wielomianu

Problem:

Małe obliczyć i wydrukować 21 wartości wielomianu:

$$y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3$$

dla

$$a_0 = 0.38461 \quad a_1 = 0.84642$$

$$a_2 = -0.32101 \quad a_3 = 0.38465$$

oraz przy x zmieniającym się od 0 do 0,05 do 1,0.

Dokładność drukowanych wyników: cztery znaki dziesiętne po kropce.

Metoda rozwiązania:

Wielomian obliczać będziemy według schematu Hornera:

$$y = ((a_3 \cdot x + a_2) \cdot x + a_1) \cdot x + a_0$$

Skala:

Ponieważ wartość bezwzględna wszystkich współczynników oraz niewiadomej są mniejsze od jedności, łatwo oszacować, że żadna z

wartości wielomianu jak i z wartości pośrednich nie przekroczy 10. Wystarczy więc przyjąć skalę 1.

Dokładność wyników:

Wobec tego, że przy skali 1 maszyna liczy z dokładnością dziesięciu znaków po kropce, pożądana dokładność czterech znaków po kropce jest łatwa do osiągnięcia.

Postać danych:

Współczynniki wielomianu umieścimy w programie, wobec czego dane wejściowe w przypadku tym nie występują.

Postać wyników:

Wyniki należy wydrukować w 21 wierszach. W każdym wierszu należy podać najpierw wartość x , a następnie wartość y , zaokrągloną do czterech cyfr po kropce, według następującego formatu

d.dd d.dddd

Odległość pomiędzy ostatnią cyfrą liczb pierwszej kolumny a znakiem liczb drugiej kolumny powinna wynosić 5 znaków dalekopicowych.

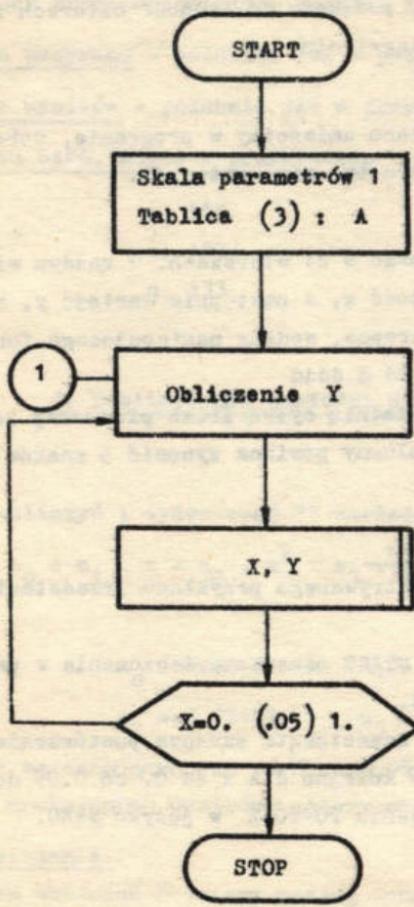
Objaśnienie sieci działań:

Sieć działań dla rozpatrywanego przykładu przedstawiona jest na rysunku 4.

Pierwsza skrzynka po START oznacza umieszczenie w programie tablicy współczynników A.

Skrzynka o kształcie sześciokąta oznacza powtórzenie wskazanej części sieci działań kolejno dla x od 0. co 0.05 do 1.0.

Skrzynka ta odpowiada zdaniu POWTÓRZ w języku SAKO.



Rysunek 4
Siec działań dla przykładu 3

Program:

K) TABLICOWANIE WARTOSCI WIELOMIANU
SKALA DZIESIETNA PARAMETROW: i
TABLICA(3):A
0.38461 0.84642
-0.32101 0.38465
*
K) OBLICZANIE Y:
USTAW SKALE DZIESIETNIE: i
CAŁKOWITE: I
* i) Y=0
* i A) Y=Y*X+A(1)
POWTORZ OD i:A:I=3(-i)o
DRUKOWANIE X,Y:
LINIA
DRUKUJ(4.2):X
DRUKUJ(6.4):Y
POWTORZ OD i:X=o.(.05)i.
STOP i
KONIEC

Objaśnienie programu:

Deklaracja SKALA PARAMETRÓW ustala skalę, w jakiej będą wprowadzane do maszyny liczby ułamkowe występujące w programie, np. współczynniki podane w TABLICY lub liczby występujące w drugim ze zdań POWTÓRZ.

Deklaracja TABLICA zapowiada wprowadzenie czteroelementowego bloku A. Elementy tego bloku są w dalszej części programu oznaczone symbolami A (I), przy czym A (0) oznacza pierwszy element. Koniec wprowadzania bloku zaznaczony jest oddzielnym wierszem z wytypowanym znakiem *.

Rozkaz USTAW SKALE powoduje przyjęcie podanej skali dla wszystkich wykonywanych po nim obliczeń.

Deklaracja CAŁKOWITE określa I jako zmienną całkowitą. Wszystkie zmienne nie wymienione w deklaracji CAŁKOWITE, traktowane są zawsze jako zmienne ułamkowe.

Obliczenie wartości Y odbywa się metodą iteracyjną. Jako war-

tość początkową Y przyjęto zero; następny wzór, powtarzony kolejno dla I równego 3, 2, 1, 0 daje w końcowym wyniku poszukiwaną wartość wielomianu dla odpowiadającej wielkości X. Powtarzanie wywołane jest zdaniem POWTÓRZ, odnoszącym się do tego wzoru.

Drukowanie X i odpowiadającemu mu Y, według wymagań podanych w punkcie 6, spowodowane jest trzema dalszymi zdaniami. W szczególności zarezerwowanych zostało 6 pozycji przed kropką liczby. Większość z nich starować będzie po prostu odstęp pomiędzy zapisem liczby X, a zapisem liczby Y.

Drugie ze zdaj POWTÓRZ powoduje powtarzanie odcinka programu od numeru '1' aż do tegoż zdania kolejno dla wartości X równych 0.00, 0.05, 0.10, ... 1.00.

Rozkaz STOP powoduje zatrzymanie się maszyny.

Deklaracja KONIEC oznacza koniec programu.

Wyniki programu:

+0.00	+0.3846
+0.05	+0.4262
+0.10	+0.4664
+0.15	+0.5056
+0.20	+0.5441
+0.25	+0.5822
+0.30	+0.6200
+0.35	+0.6580
+0.40	+0.6964
+0.45	+0.7355
+0.50	+0.7756
+0.55	+0.8170
+0.60	+0.8600
+0.65	+0.9048
+0.70	+0.9517
+0.75	+1.0011
+0.80	+1.0532
+0.85	+1.1084
+0.90	+1.1668
+0.95	+1.2288
+1.00	+1.247

Nowe pojęcia SAKO użyte w przykładzie:

Blok liczbowy	s. 97, 115
TABLICA	s. 117
CAŁKOWITE	s. 120
SKALA PARAMETRÓW	s. 120
POWTÓRZ	s. 124

4. Mnożenie macierzy kwadratowej

przez macierz względem niej przedstawioną

Problem:

Dana jest macierz kwadratowa B o ilości elementów nie większej niż 12. Należy ułożyć program obliczenia i wydrukowania macierzy A wyznaczonej iloczynem macierzowym

$$[A] = [B] \cdot [B]'$$

gdzie $[B]'$ oznacza macierz przedstawioną w stosunku do B. Macierz $[A]$ będziemy nazywali dalej krótko macierzą wynikową.

Metoda rozwiązania:

Program podzielimy na trzy rozdziały:

ROZDZIAŁ: 1 - Wprowadzenie macierzy kwadratowej,

ROZDZIAŁ: 2 - Obliczenie macierzy wynikowej,

ROZDZIAŁ: 3 - Wydawnictwo macierzy kwadratowej.

Podział taki ma następujące zalety:

- Powyższe rozdziały mają charakter niezależny i mogą być w całości użyte przy rozwiązywaniu innych podobnych problemów.
- Ponieważ program został podzielony na trzy części, ilość miejsc pozostawionych w pamięci wewnętrznej na pomieszczenie współczynników macierzy w każdym rozdziale jest większa.

Obliczenia macierzy wynikowej dokonamy według wzoru

$$a_{ij} = \sum_{k=0}^{N-1} b_{ik} b_{jk}$$

gdzie N oznacza ilość elementów wiersza lub kolumny. Należy zawsze pamiętać, że w języku SAKO indeksowanie rozpoczynamy zawsze od zera, a nie od 1. Pierwszym elementem macierzy jest więc A (0,0) a nie A (1,1), ostatnim natomiast A (N-1, N-1).

Skala:

Załóżmy, że wszystkie elementy macierzy B i A są mniejsze od 10, natomiast żaden z wyników pośrednich lub końcowych nie przekracza 1000. Wystarczająca jest zatem skala 3.

Dokładność wyników:

Załóżmy, że pożądane są 4 cyfry znaczące po kropce dziesiętnej. Ponieważ w skali 3 maszyna liczy z dokładnością 7 znaków dziesiętnych po kropce, żądana dokładność będzie osiągnięta.

Postać danych:

Przyjmujemy postać danych wejściowych, wynikającą z rysunku 5. W pierwszym wierszu formularza umieszczamy komentarz WSPÓŁCZYNNIKI MACIERZY B, który nie jest uwzględniany przy czytaniu danych.

W drugim wierszu formularza umieszczamy najpierw komentarz "RZAD MACIERZY=", który również nie jest uwzględniany przy czytaniu danych przez maszynę, a dalej liczbę N, np. 10, mówiącą o rozędzie tej macierzy.

Następnie wprowadzamy kolejne wiersze macierzy. Każdy wiersz rozpoczynamy komentarzem, np. "W7:", gdzie liczba oznacza numer wiersza /tutaj numerację rozpoczynamy od 1/. Następnie w każdym wierszu formularza wypisujemy po pięć elementów wiersza macierzy aż do jego wyczerpania.

Po wypisaniu wszystkich elementów macierzy zgodnie z obowiązującymi regułami pozostawiamy jeden wiersz wolny z napisanym jedynie znakiem *.

WSPÓŁCZYNNIKI MACIERZY B,

RZĄD MACIERZY = 8

W 1: _____

W 2: _____

...
W 8: _____

*

Rysunek 5

Postać danych i wyników w przykładzie 4 dla N = 8

Postać wyników:

Żądamy, aby wyniki były wydane w tej samej postaci, co dane wejściowe, aby mogły służyć bez przepisywania jako dane wejściowe w innych programach.

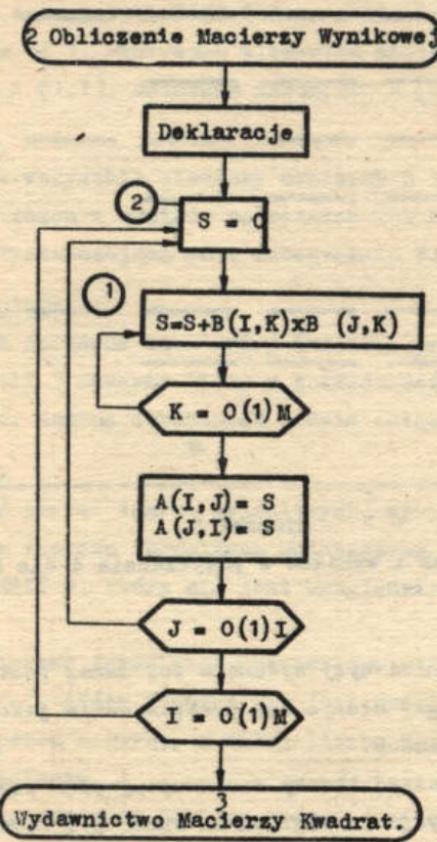
Przed kropką każdej liczby rezerwujemy pięć pozycji /dwie spacje oraz trzy pozycje ze względu na skalę 3/. Liczby zaokrąglamy do czterech cyfr po kropce.

Sieci działań:

Sieci działań dla tego przykładu podane są na rysunkach 6 oraz 7.

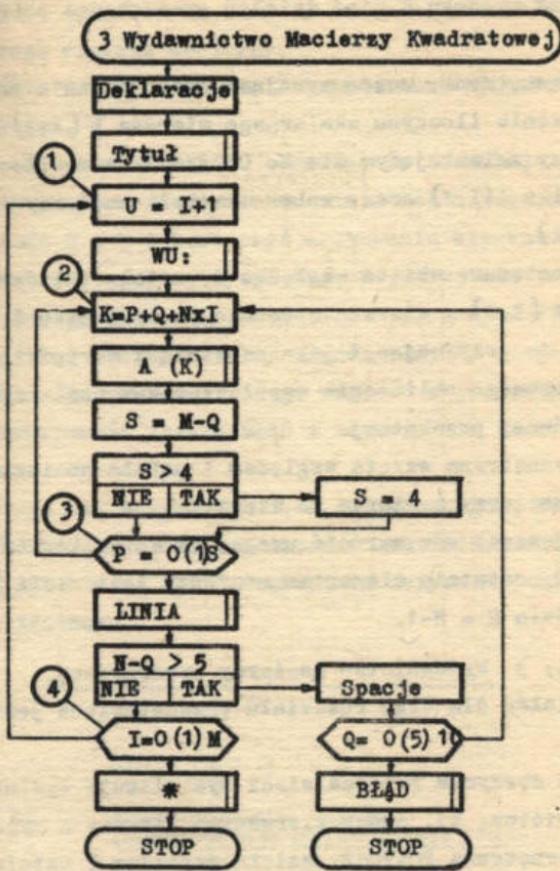
Objaśnienie sieci działań:

ROZDZIAŁ: 1 Wprowadzenie macierzy kwadratowej.



Rysunek 6

Sieć działań dla rozdziału: 2
 ("Obliczanie macierzy wynikowej")



Rysunek 7

Siedć działań dla rozdziału: 3
"Wydawnictwo Macierzy Kwadratowej"

W przypadku tym struktura programu jest tak prosta, że sieć działań jest zbyteczna.

ROZDZIAŁ: 2 Obliczenie macierzy wynikowej.

Podana na rysunku 6 sieć działań przedstawia potrójną pętlę POWTÓRZ.

Pętla wewnętrzna, rozpoczynająca swoje działanie przy $S=0$ powoduje obliczenie iloczynu skalarnego wiersza $B(I, K)$ przez wiersz $B(J, K)$ przy zmieniającym się K . Otrzymany sumo-iloczyn daje nam współczynniki $A(I, J)$ oraz, wobec symetrii macierzy A , współczynniki $A(J, I)$.

Pętla następna, wzięta względem J , ustala obliczenie współczynników $A(I, J)$ w wierszu o ustalonym wskaźniku I . Należy zwrócić uwagę, że przyjmując I jako ostatnią z wartości J unikamy tym samym zbytecznego obliczenia współczynników macierzy po drugiej stronie głównej przekątnej.

Pętla zewnętrzna wzięta względem I ustala obliczenie współczynników macierzy A wiersz po wierszu.

Należy jeszcze raz zwrócić uwagę, że wobec konwencji przyjętych w SAKO, ostatnim elementem macierzy jest nie $A(N, N)$ lecz $A(M, M)$, gdzie $M = N-1$.

ROZDZIAŁ: 3 Wydawnictwo macierzy kwadratowej.

Siec działań dla tego rozdziału przedstawiona jest na rysunku 7.

Pierwsza skrzynka robocza sieci symbolizuje wydrukowanie TYTUŁU wydawnictwa, tj. dwóch pierwszych wierszy z rysunku 5.

Pętla zewnętrzna POWTÓRZ, wzięta względem I ustala drukowanie kolejnych wierszy macierzy, poczynając od $I = 0$.

Ponieważ w postaci zewnętrznej pierwszy wiersz zaznaczamy jako W1 a nie jako W0, zachodzi konieczność drukowania przed każdym wierszem WU gdzie $U = I + 1$.

Ponieważ w rozkazie DRUKUJ można używać zmiennej najwyższej jednoindeksowej, ustalamy indeks K dla współczynnika macierzy w postaci $A(K)$ przez równanie

$$K = P + Q + N \cdot I,$$

gdzie $0 \leq P \leq 4$, Q zaś jest wielokrotnością 5. W ten sposób P przedstawia pozycję w wierszu arkusza, natomiast liczba Q/5 przedstawia ilość dodatkowych wierszy arkusza, potrzebnych dla zapisania jednego wiersza macierzy.

Pętla POWTÓRZ wzięta względem P przedstawia drukowanie jednego wiersza arkusza.

Spełnienie warunku $N - Q > 5$ oznacza, że pozostały jeszcze dalsze elementy macierzy do wydrukowania i wobec tego należy zwiększyć wartość Q o 5 i powtórzyć drukowanie wiersza arkusza. Dodatkowe SPACJE zastępują pozycje zajęte pierwotnie przez WU:..

Przypadek, gdy przy $Q = 10$ jest $N - Q > 5$, oznaczałby, że $N > 15$, a więc rząd macierzy byłby większy od 12, co jest wykluczone w założeniach programu. Z tego względu w przypadku tym przewidujemy drukowanie wyrazu BŁĄD z odpowiednim komentarzem oraz zatrzymanie się maszyny.

Po wyczerpaniu się wszystkich I, w odpowiadającej tej zmiennej pętli POWTÓRZ drukujemy dodatkowo w oddzielnym wierszu znak *, tak, aby wydrukowane wyniki spełniały wszystkie warunki stawiane danym wejściowym.

Program:

ROZDZIAŁ: 1

K) WPROWADZENIE DANYCH
 USTAW SKALE DZIESIETNIE: 3
 CALKOWITE: N,M
 BLOK(0):N,M
 BLOK(1,1): A, B
 CZYTAJ:N
 M=N-1
 STRUKTURA(M,M): A,B
 CZYTAJ:•B
 IDZ DO ROZDZIAŁU: 2

ROZDZIAŁ: 3

- K) WYDAWNICTWO MACIERZY WYNIKOWEJ
CALKOWITE: N,M,U,I,J,K,P,Q,S
BLOK(o): N,M
BLOK(i43): A
STRUKTURA(M,M): A
K) DRUKOWANIE TYTULU
TEKST WIERSZY 1:
WYDAWNICTWO MACIERZY A
TEKST:
RZAD MACIERZY=
DRUKUJ(3):N
LINIA
• 1) U=1+1
K) DRUKOWANIE WU:
TEKST:
W
DRUKUJ(2):U
TEKST:
:
•• 2) K=P+Q+N×I
DRUKUJ(5+4):A(K)
S=M-Q
GDY S>4: NASTEPNY, INACZEJ 3
S=4
3) POWTORZ OD z:P=o(1)s
LINIA
GDY N-Q>5: NASTEPNY, INACZEJ 4
SPACJI 5
POWTORZ OD z:Q=o(5)10
LINIA
TEKST WIERSZY 1:
BLAD. ILOSC ELEMENTOW Wiersza WIEKSZA OD 15
STOP 1
4) POWTORZ OD z:I=o(1)m
SPACJI 20
TEKST:
•
LINIA 10
STOP NASTEPNY
KONIEC

ROZDZIAŁ: 2

K) OBLICZENIE MACIERZY WYNIKOWEJ

CALKOWITE: N, M, K, I, J

BLOK(o): N, M

BLOK(iI, iI): A, B

STRUKTURA(M, M): A, B

*** S=o

* i) S=S+B(I,K)*B(J,K)

POWTORZ OD i:K=o(i)M

A(I,J)=S

A(J,I)=S

POWTORZ OD z:J=o(i)I

POWTORZ OD z:I=o(i)I

IDZ DO ROZDZIAŁU: 3

Wyniki działania programu dla macierzy jednostkowej 6-go rzędu:

WYDAWNICTWO MACIERZY A

RZAD MACIERZY=	6				
W 1:	+1.0000	+0.0000	+0.0000	+0.0000	+0.0000
	+0.0000				
W 2:	+0.0000	+1.0000	+0.0000	+0.0000	+0.0000
	+0.0000				
W 3:	+0.0000	+0.0000	+1.0000	+0.0000	+0.0000
	+0.0000				
W 4:	+0.0000	+0.0000	+0.0000	+1.0000	+0.0000
	+0.0000				
W 5:	+0.0000	+0.0000	+0.0000	+0.0000	+1.0000
	+0.0000				
W 6:	+0.0000	+0.0000	+0.0000	+0.0000	+0.0000
	+1.0000				

*

Objaśnienie programu:

Cały program, zgodnie z założeniami, składa się z trzech rozdziałów. Każdy z nich zatytułowany jest deklaracją ROZDZIAŁ, aczkolwiek w stosunku do pierwszego rozdziału nie jest to w tym przypadku konieczne.

Ostatnim wykonywanym zdaniem pierwszego i drugiego rozdziału jest zdanie IDZ DO ROZDZIAŁU. Ostatnim wykonywanym zdaniem trzeciego rozdziału jest zdanie STOP. Ostatnim wypisanym zdaniem języka SAKO jest deklaracja KONIEC, oznaczająca zakończenie programu.

ROZDZIAŁ: 1 Wprowadzenie macierzy kwadratowej rozpoczęyna się od zarezerwowania 289 miejsc pamięci: po jednym na liczby całkowite N i M oraz 2 razy po 144 (= 288) miejsc na liczby ułamkowe bloków A i B. Deklaracja STRUKTURA określa bloki A i B jako macierze kwadratowe rzędu N. Liczba N przeczytana będzie uprzednio przez program i nie może być większa od 12. Deklaracja STRUKTURA nie rezerwuje już żadnych miejsc pamięci; byłoby to zresztą niemożliwe, gdyż w trakcie czytania programu nie jest jeszcze znana liczba N.

Zdanie CZYTAJ powoduje przeczytanie $N \times N$ wartości bloku B.

Ostatnie zdanie tego rozdziału powoduje przejście do rozdziału wskazanego. Zostanie on automatycznie wpisany do pamięci wewnętrznej i rozpoczęcie się jego wykonywanie począwszy od pierwszego zdania.

ROZDZIAŁ: 2 Obliczenie macierzy wynikowej rozpoczęyna się również od zarezerwowania miejsc pamięci na liczby N i M oraz bloki A i B. W miejscach tych rozdział niniejszy znajduje wartości pozostawione przez rozdział poprzedni. Nazwane one zostały, podobnie jak poprzednio, symbolami N, M oraz A i B,aczkolwiek możliwe byłoby przyjęcie i innych nazw.

Obliczenie macierzy A następuje zgodnie z załączoną siecią działań.

ROZDZIAŁ: 3 Wydawnictwo macierzy kwadratowej znajduje rząd wydawanej macierzy w pierwszym zarezerwowanym miejscu, oraz wydawaną macierz w miejscach, które zostały nazwane, podobnie jak poprzednio, symbolami N oraz A.

Drukowanie TYTUŁU odbywa się następująco. Pierwszy wiersz wy-

drukowany jest w całości przez rozkaz TEKST WIERSZY, po którym następuje automatyczne przejście do drukowania następnego wiersza. Pierwsza część drugiego wiersza wydrukowana jest przez zdanie TEKST. Bezpośrednio po tym drukujemy liczbę N, wskazującą rząd opracowywanej macierzy.

Podobnie do drugiego wiersza odbywa się drukowanie komentara WU:

Dalszy ciąg programu jest zgodny z zakońzoną siecią działań.

Sprawdzenie programu:

Prawidłowość ułożenia rozdziału pierwszego i trzeciego możemy sprawdzić następująco:

Bierzemy pewną macierz przy N równym na przykład 6. Wprowadzamy ją do maszyny przy pomocy rozdziału pierwszego. Przy pomocy rozdziału drugiego, ułożonego specjalnie w tym celu, przypisujemy tylko tę macierz z bloku B do bloku A, zgodnie z oznaczeniami w programie. Rozdział trzeci wydaje rozpatrywaną macierz. Postać wyników powinna być identyczna z postacią danych wejściowych.

Sprawdzenie części programu można przeprowadzić, wykonując ten program dla pewnej macierzy i sprawdzając kilka wyników. W szczególności macierz wynikowa powinna być symetryczna.

Sprawdzenie wyników:

Po każdym wykonaniu programu wyniki można sprawdzić przez skontrolowanie kilku wartości otrzymanych wyników.

Nowe pojęcia SAKO użyte w przykładzie:

ROZDZIAŁ	s. 112
STRUKTURA	s. 117
IDZ DO ROZDZIAŁU	s. 132
TEKST	s. 148
TEKST WIERSZY	s. 149
SPACJA	s. 150

5. Obliczanie pierwiastka sumy kwadratów

w zmiennej skali

Problem:

Należy ułożyć program obliczenia pierwiastka z sumy kwadratów liczb danych na wejściu. Ilość tych liczb nie przekracza 100. W celu uzyskania możliwie dokładnych wyników, obliczenie powinno być przeprowadzone w możliwie niskiej skali.

Metoda rozwiązania:

Kolejne wczytywanie liczb danych i tworzenie sumy. Na końcu wyciągnięcie pierwiastka kwadratowego.

Dla zwiększenia przejrzystości programu i możliwości wprowadzenia większego bloku liczbowego program napiszemy w dwu rozdziałach:

ROZDZIAŁ: 1 Wczytywanie danych

ROZDZIAŁ: 2 Obliczenie pierwiastka z sumy kwadratów.

Skala:

Za każdym razem wraz z danymi należy podać skalę, w której można wprowadzić dane i w której obliczenie przypuszczalnie da się jeszcze wykonać. Jeżeli w czasie obliczeń wystąpi nadmiar, należy obliczenie powtórzyć w wyższej skali i ponawiać je aż do przypadku, gdy nadmiar już nie wystąpi.

Dokładność wyniku:

W skali S maszyna liczy z dokładnością 10-S znaków po przecinku. Licząc, że błędy zaokrągleń nie przewyższają dwu znaków dziesiętnych, otrzymujemy pewnych 8-S znaków po przecinku, do których zaokrąglając będziemy wyniki.

Postać danych:

Jako pierwsza jest wydrukowana liczba S równa skali w jakiej będziemy wykonywać obliczenia, następnie liczba N, wskazująca ilość liczb wejściowych / $N \leq 100$ /, po czym wydrukowany jest blok tych liczb.

Postać wyniku:

Pojedyncza liczba, zaokrąglona do odpowiedniej ilości znaków dziesiętnych.

Sieć działań:

Sieć działań dla tego przykładu podana jest na rysunku 8.

Objaśnienie sieci działań:

Obliczenie przeprowadzamy najpierw według przyjętej pierwotnej skali działań. W przypadku nadmiaru zwiększymy skalę S o jeden oraz przeskalowujemy o jeden cały blok A. Następnie znów przeprowadzamy obliczenie, badamy czy był nadmiar itd. W końcu dochodzimy do skali S, w której nadmiar już nie występuje. Wówczas obliczamy pierwiastek, przygotowujemy drukowanie wyniku przepodanie właściwej ilości pozycji przed i po kropce dziesiętnej, a na końcu drukujemy wynik.

Przejście ze skrzynki 'USTAW SKALE' do skrzynki 'OBLCZENIE SUMY Q' zaznaczono za pomocąłącznika z wpisana jako symbolem liczbę '1'.

Program:

- K) OBLCZANIE PIERWIASTKA Z SUMY KWADRATOW
ROZDZIAŁ: 1
K) CZYTANIE DANYCH
CALKOWITE: S,N,M
CZYTAJ: S,N
USTAW SKALE DZIESIETNIE: S
M=11-1
BLOK(99): A
STRUKTURA(M):
CZYTAJ: *A
IDZ DO ROZDZIAŁU: 2
- K) ROZDZIAŁ: 2
OBLCZENIE
CALKOWITE: K,M,S,U,V,N
BLOK(0): S,N,M
BLOK(99): A
STRUKTURA(M): A

d. c.

1) $Q=0$
• 1A) $Q=Q+A(K)*z$
POUTORZ OD 1A: $K=o(1)N$
GDY BYŁ NADMIAR: NASTEPNY, INACZEJ 2
 $S=S+z$
ZWIĘKSZ SKALE DZIESIETNIE O 1: *A
USTAW SKALE DZIESIETNIE:S
SKOCZ DO 1
2) $P=PWK(Q)$
K) PRZYGOTOWANIE DRUKOWANIA
 $U=S+3$
 $V=8-S$
LINIA
DRUKUJ(U,V): P
STOP NASTEPNY
IDZ DO ROZDZIAŁU: 1
KONIEC

Objaśnienia programu:

Program w rozpatrywanym przykładzie napisany został bez trudu na podstawie sieci działań. Należy zwrócić uwagę na przygotowywanie zdania DRUKUJ.

Sprawdzenie programu:

Program sprawdza się przez wykonanie programu dla szczególnego przypadku, dającego znany wynik.

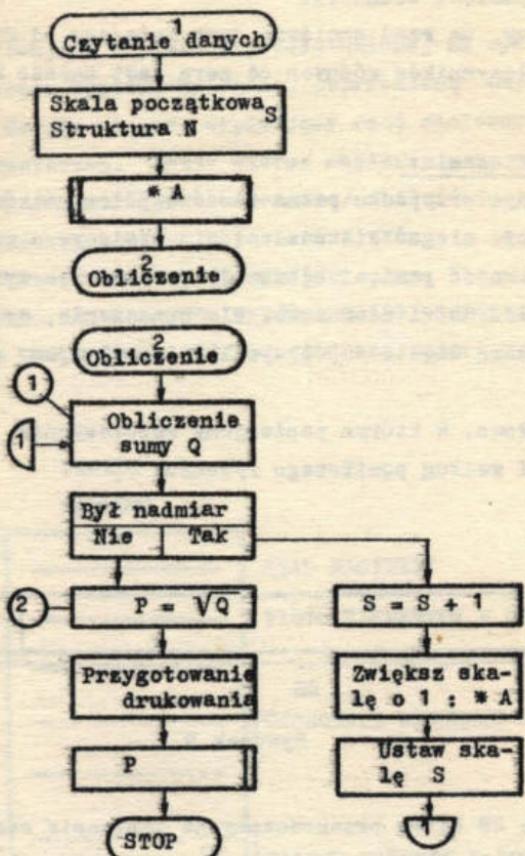
Sprawdzenie wyniku:

W powyższym przykładzie sprawdzenie wyniku możliwe jest tylko przez powtórzenie obliczenia. Metodę tę stosujemy często w przypadkach, gdy czas wykonywania programu nie jest długi.

Nowe pojęcia SAKO występujące w tym przykładzie:

GDY BYŁ NADMIAR s. 134

ZWIĘKSZ SKALE DZIESIETNIE s. 141



Rysunek 8

Sieć działań dla przykładu 5

6. Cechowanie współczynników macierzy

Problem:

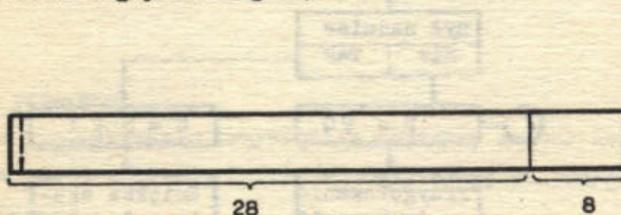
Należy ułożyć rozdział programu, wczytujący współczynniki macierzy do pamięci bębornej.

Zakładamy, że rząd macierzy jest mniejszy od 255, przy czym ilość współczynników różnych od zera jest zawsze mniejsza od 4.000.

Metoda rozwiązania:

W skrajnym przypadku pełna ilość współczynników rozpatrywanych macierzy może sięgać kilkudziesięciu tysięcy, a więc więcej niż wynosi pojemność pamięci bębornej. Należy więc wpisywać do niej tylko współczynniki niezerowe. Dla oznaczenia, do jakich niewiadomych odnoszą się te współczynniki, przyjmujemy następującą metodę.

Każde słowo, w którym zapisujemy współczynnik, dzielimy na dwie części według poniższego rysunku:



Rysunek 9

Pierwsze 28 bitów przeznaczamy na zapisanie samego współczynnika. Pozostałe 8 bitów przeznaczamy na zapis wskaźnika niewiadomej, przy którym występuje współczynnik. Bitы te pozwalają na wyróżnienie $2^8 - 1 = 255$ wskaźników, nie licząc wskaźnika zerowego.

Skala:

Skalę wprowadzania współczynników wprowadzamy wraz ze współczynnikami.

³ Paragraf ten można opuścić przy pierwszym czytaniu podręcznika.

Dokładność wyników:

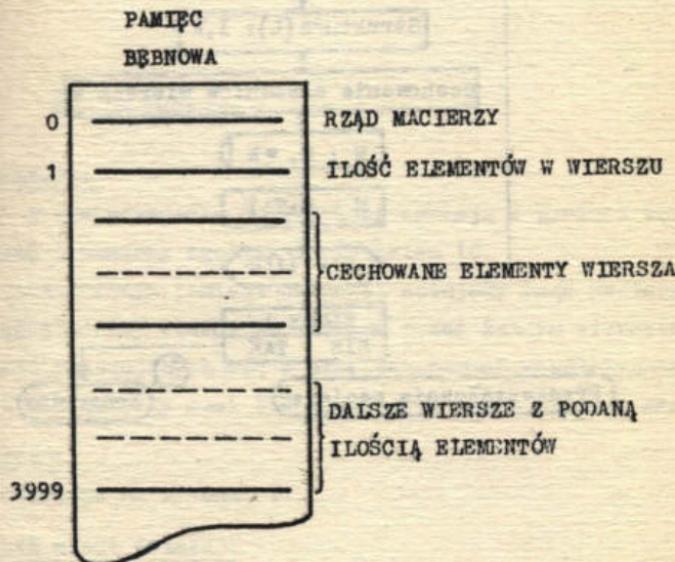
Liczba 28 bitów przeznaczonych na każdy element macierzy pozwala osiągnąć dokładność przedstawienia tego elementu w pamięci maszyny równoważną około 8 znakom dziesiętnym.

Postać danych:

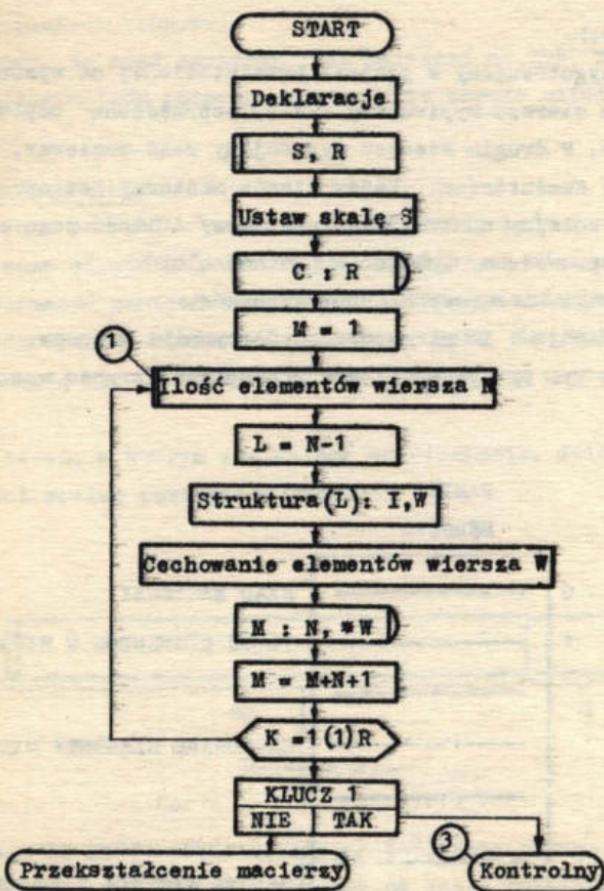
Dane przygotowujemy w postaci przedstawionej na rysunku 10. W pierwszym wierszu wypisujemy skalę, poprzedzoną odpowiednim komentarzem. W drugim wierszu wypisujemy rząd macierzy, również poprzedzany komentarzem. Każdy wiersz macierzy rozpoczynamy podając jego kolejny numer /jako komentarz/ i ilość jego elementów. Następnie wprowadzamy kolejno w postaci bloków:

- wskaźniki niezerowych współczynników,
- odpowiadające im niezerowe współczynniki wiersza.

Użycie w tym przypadku bloków przyspiesza proces wprowadzania danych.



Rysunek 10
Rozmieszczenie macierzy w pamięci bębnowej



Rysunek 11

Sięć działań dla przykładu 6

Postać danych wejściowych:

SKALA = 2

RZAD MACIERZY = 130

W₁ ELEMENTOW: 13

1	3	4	12	13
30	30	40	62	70
110	118			

*

3.84	2.38	0.14	-1.10	-0.11
1.12	-3.18	-0.78	-2.01	1.08
4.13	-6.03			

*

W₂ ELEMENTOW: 18

Postać wyników:

Wyniki w tym przypadku umieszczone zostają w pamięci bębnowej. Ich rozkład planujemy zgodnie z rysunkiem 10.

W miejscu zerowym pamięci bębnowej notujemy rzad macierzy. Następnie wprowadzamy wiersz po wierszu przed każdym wierszem wpisując ilość znajdujących się w nim cechowanych współczynników. Wpisana ilość wierszy musi być oczywiście równa rzędowi macierzy.

Siedz działań:

Przedstawiona na rysunku 11.

Objaśnienie siedzi działań:

Zmienna M służy dla ustalenia miejsca na bębnie, od którego każdorazowo wpisujemy zmienną N i blok W.

Program:

K) ROZDZIAŁ: 1
CECHOWANIE WSPOLCZYNNIKOW MACIERZY
CALKOWITE:S,R,M,N,L,I,K,P,
BLOK(o): M,N,L
CZYTAJ:S,R
USTAW SKALE DZIESIETNIE:S
PISZ NA DEBEN OD o:R
M=1
BLOK(49):I,W
•1) CZYTAJ:N
L=N-1
STRUKTURA(L):I,W
CZYTAJ:•I,•W
K) CECHOWANIE WSPOLCZYNNIKOM WERSZA W
•2) A=U(K)
B=I(K)
A=((A*777*777*777*400)*18+B)*(-18)
W(K)=A
POUTORZ OD z:K=o(1)L
PISZ NA BEBEN OD M:N,•W
i=N+N+1
POUTORZ OD i:P=i(1)R
GDY KLUCZ i:3, NACZEJ NASTEPNY
K) PRZEJSIE DO PRZEKSZTALCANIA MACIERZY
IDZ DO ROZDZIAŁU: z
3) IDZ DO ROZDZIAŁU: 3

Objaśnienia programu:

Cechowanie każdego ze współczynników $W(k)$ odbywa się przy pomocy formuły bulowskiej, odróżnionej znakiem \equiv od formuły arytmetycznej.

W stałej bulowskiej

$$777.777.777.400$$

pierwszych 28 pozycji stanowią same jedynki, a pozostałych 8 pozycji - same zera. Pomnożenie logiczne $U(k)$ ($= A$) przez tę stałą pozostawia więc pierwszych 28 pozycji $W(k)$ bez zmiany, zeruje natomiast 8 ostatnich jej pozycji.

Wskaźniki $I(k)$ ($= B$) jako liczby całkowite zajmują skrajnie

prawe pozycje słowa krótkiego, które zawsze stanowią lewą połowę słowa długiego.

Przesunięcie w prawo wskaźnika $I(k)$ o osiemnaście pozycji przekonosi go w skrajnie prawą pozycję słowa długiego.

Suma logiczna składników uzyskanych w podany powyżej sposób daje współczynnik ocechowany zgodnie z założeniami. Oznaczamy go ponownie jako $W(k)$.

Sprawdzenie programu:

Sprawdzenie powyższego programu dokonać można przy pomocy specjalnego programu kontrolnego, sprawdzającego poprawność zapisania macierzy na bęben przy odpowiednim jej dobraniu.

Przy normalnym wykonywaniu programu klucz 1, znajdujący się na stoliku operatora, jest opuszczony i po wprowadzeniu macierzy na bęben następuje przejście do rozdziału 2: PRZEKSZTAŁCENIE MACIERZY. Przy kontroli programu klucz 1 jest podniesiony i po wprowadzeniu macierzy następuje przejście do rozdziału 3: KONTROLNY.

Korzystanie z kluczy na stoliku operatora dla włączenia dodatkowych operacji czy całych rozdziałów kontrolnych należy do typowych ich zastosowań.

Innym przykładem korzystania z kluczy są programy posiadające kilka zbliżonych wariantów wykonania. Przez wprowadzenie zdai GDY KLUCZ i odpowiednie ustawienie kluczy możemy wybrać jeden z możliwych wariantów.

Sprawdzenie wyników:

Problem sprawdzenia wyników dla tego rozdziału należy rozpatrywać łącznie z pozostałymi rozdziałami programu.

Nowe pojęcia SAKO:

GDY KLUCZ s. 135

Formuła Bulowska s. 137

7. Analiza wiersza tytuловego⁴

Problem:

Załóżmy, że mamy ułożyć program dotyczący na przykład odwracania macierzy. Program ten może ulec znaczнемu uproszczeniu, o ile odwracana macierz jest symetryczna. Ponieważ jest to bardzo łatwe do stwierdzenia przed wprowadzeniem danych do maszyny, wobec tego w pierwszym wierszu danych wejściowych dajemy jedną z dwu informacji:

MACIERZ SYMETRYCZNA

MACIERZ NIESYMETRYCZNA

Rozdział analizy tytułu po przeczytaniu tego wiersza powinien spowodować przejście do jednego z dwóch rozdziałów:

ROZDZIAŁ: 2

ROZDZIAŁ: 3

które zawierać mogą odpowiednie programy dla odwracania macierzy symetrycznej lub niesymetrycznej.

Ponadto, informacja dotycząca rodzaju macierzy powinna być wypisana na wyjściu.

Metoda rozwiązania:

Rozdział analizy tytułu pobiera pierwszy wiersz danych wejściowych i bada znak po znaku, licząc od lewej do prawej. Jeżeli z dwóch znaków S i N pierwszym napotkanym jest S - macierz jest symetryczna, jeżeli zaś N - niesymetryczna.

Wartością litery S jest 51, litery N jest 46⁵.

Skala:

W przykładzie tym nie ma liczb ułamkowych.

Dokładność wyników:

W przykładzie tym nie ma liczb ułamkowych.

Postać danych:

Podana w punkcie 1 przykładowo.

⁴ Paragraf ten można opuścić przy pierwszym czytaniu podręcznika.

⁵ Wartości liczbowe liter podane są w rozdziale I, tabela 1.

Postać wyników:

Przejście do jednego z dwu rozdziałów.

Sieć działań:

Przedstawiona na rysunku 12.

Objaśnienia sieci działań:

Przedstawiona sieć działań jest prostą realizacją przyjętej metody rozwiązania. Po przeczytaniu wiersza i umieszczeniu go w pamięci jako wektora W następuje badanie znaku po znaku i (w zależności od wyniku) przejście do jednego z dwu rozdziałów. W wektorze W przewidziano 70 składowych, co przekracza ilość znaków, jaka może być zapisana w jednym wierszu na dalekopisie. W przypadku nienapotkania żadnego z poszukiwanych znaków następuje sygnaлизacja błędu przez wydrukowanie słowa BLAD.

Program:

```

ROZDZIAL: 1
K) ANALIZA TYTULU
    CAŁKOWITE: W, 1
    BLOK(69): W
    rA) CZYTAJ WIERSZ: W
        DRUKUJ WIERSZ: W
        •1) GDY W(1)=51: NASTEPNY, INACZEJ 2
            IDZ DO ROZDZIAŁU: 2
    K) MACIERZ SYMETRYCZNA
        •2) GDY W'1)=46: NASTEPNY, INACZEJ 3
            IDZ DO ROZDZIAŁU: 3
    K) MACIERZ NIESYMETRYCZNA
        •3) POWTORZ OD 1: I=o(1)69
            TEKST WIERSZY 1:
            BLAD
            STOP rA
            KONIEC

```

Objaśnienie programu:

Dwie pierwsze deklaracje określają wektor W jako ciąg 70 liczb całkowitych. Przyjęcie "wektora całkowitego" nie tylko daje

oszczędność na miejscach pamięci, ale również pozwala traktować różne znaki jako odpowiadające im liczby całkowite.

Zdanie CZYTAJ WIERSZ powoduje wpisanie wszystkich znaków wiersza jako elementów składowych wektora W .

Zdanie DRUKUJ WIERSZ powoduje wydrukowanie przeczytanego wiersza na formularzu wyników, zgodnie z wymaganiem postawionym w pozycji PROBLEM.

Dalsze zdanie powoduje badanie znaków wiersza zgodnie z przedstawioną siecią działań.

Sprawdzenie programu:

Sprawdzenia można dokonać przez próbę. Jako rozdziałów SYMETRYCZNA I NIESYMETRYCZNA należy użyć rozdziałów o nazwach 2 i 3, sygnalizujących prawidłowość przejścia.

Sprawdzenie wyników:

W przedstawionym przykładzie wyników liczbowych nie ma.

Nowe pojęcia użyte w przykładzie:

CZYTAJ WIERSZ s. 144

DRUKUJ WIERSZ s. 147

8. Obliczenie wartości funkcji

Problem:

Niech WP (A , B , C) oznacza Większy Pierwiastek równania kwadratowego

$$A \cdot x^2 + B \cdot x + C = 0 \quad (B^2 - 4AC > 0)$$

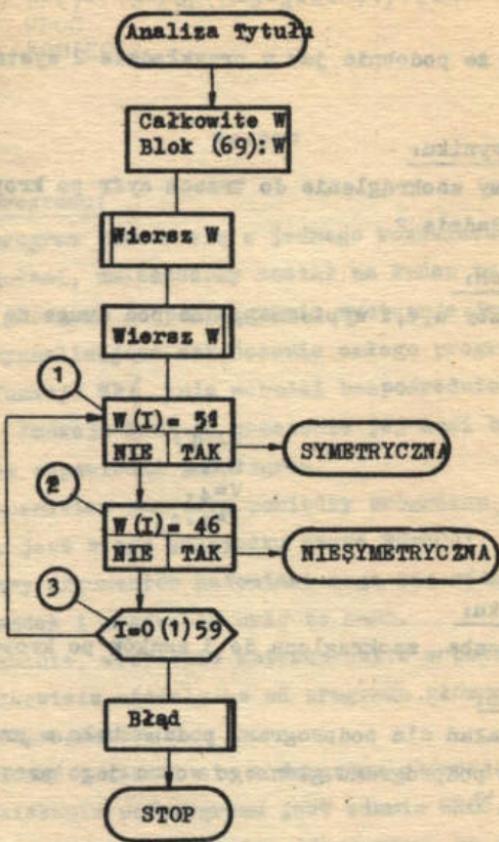
Należy obliczyć funkcje

$$Z = u + v \cdot WP(u, u + v, t) \cdot WP(u, u - v, t),$$

gdzie u, v, t , są to trzy liczby zadane, spełniające warunki

$$(u + v)^2 - 4 \cdot u \cdot t > 0$$

$$(u - v)^2 - 4 \cdot u \cdot t > 0$$



Rysunek 12
Sieć działań dla przykładu 7

Metoda rozwiązania:

W programie posłużymy się podprogramem obliczenia Większego Pierwiastka, ułożonym na podstawie przykładu 2.

Skala:

Założmy, że podobnie jak w przykładzie 2 wystarczającą jest skala 3.

Dokładność wyniku:

Przyjmiemy zaokrąglenie do trzech cyfr po kropce, podobnie jak w przykładzie 2.

Postać danych:

Trzy liczby u, v, t wypisane jedna pod drugą na formularzu wejściowym.

DANE:

$U=1.$

$V=4.$

$T=2.25$

Postać wyniku:

Jedna liczba, zaokrąglona do 3 znaków po kropce.

Siec działań:

Siec działań dla podprogramu podana była w przykładzie 2. Sieć działań dla podprogramu głównego wobec jego prostoty może być opuszczona.

Objaśnienie sieci działań:

Patrz: Sieć działań.

Program:

```
K} OBLCZANIE WARTOSCI FUNKCJI  
i} USTAN SKALE DZIESIETNIE: 3  
    CZYTAJ: U,V,T  
    Z=U+V*WP(U,V+U,T)*WP(U,V-U,T)  
    LINIA  
    DRUKUJ(4.3): Z  
    STOP i
```

PODPROGRAM: WP(A,B,C)
GDY A>0: NASTEPNY, INACZEJ I
WP()=(-B+PWK(B*2-4*A*C))/(2*A)
WRÓC
1) WP()=(-B-PWK(B*2-4*A*C))/(2*A)
WRÓC
KONIEC

Wynik:

+4.000

Objaśnienie programu:

Powyższy program składa się z jednego rozdziału. Podprogram, zgodnie z regułami, umieszczony został na końcu tego rozdziału po programie głównym. Po podprogramie występuje jedynie deklaracja KONIEC, sygnalizująca zakończenie całego programu.

Ponieważ funkcja WP() nie wchodzi bezpośrednio w skład języka SAKO, jak np. funkcja SIN(), znaczenie jej musi być dodatkowo określone przez odpowiedni podprogram.

Jednym łącznikiem nazwowym pomiędzy programem głównym, a jego podprogramem, jest w tym przypadku nazwa FUNKCJI, w tym przypadku nazwa WP. Nazwy argumentów natomiast mogą się różnić między sobą, byle ich porządek i znaczenie było to samo.

Ujmując ogólnie, wszystkie zmienne użyte w podprogramie mają znaczenie całkowicie niezależne od programu głównego i ewentualnie innych podprogramów.

Skala obliczeń przenosi się z programu głównego do podprogramu. Koncem działania podprogramu jest zdanie WRÓĆ.

Konsekwencje wynikające z użycia podprogramu, są następujące:

- Pomimo, że funkcja WP() występuje w formule określającej z dwukrotnie i za każdym razem z innym argumentem, podprogram na obliczenie tej funkcji występuje tylko raz.
- Podprogram może być uprzednio sprawdzony i przechowywany w gospodowej postaci w bibliotece podprogramów.

Sprawdzenie podprogramu:

Przez wykonanie programu dla wartości, dla których wyniki są znane.

Prawdzenie wyników:

Przez dwukrotne wykonanie programu.

Nowe pojęcia SAKO, występujące w tym przykładzie:

Funkcje	s. 102
PODPROGRAM	s. 113
WRÓĆ	s. 132

9. Tablicowanie pierwiastków funkcji przestępnej

Problem:

Należy ułożyć tablicę pierwiastków równania:

$$\sin(\pi/2 \cdot x) - \alpha \cdot x = 0$$

spełniających nierówność $1 \leq x \leq 2$ w zależności od parametru α , zmieniającego się od 0 do 1 co 0.05.

Metoda rozwiązań:

W programie posłużymy się podprogramem, który nazwiemy PMS /Pierwiastek metodą siecznych/ i który dla dowolnej funkcji $F()$ pozwala znaleźć pierwiastek równania $F(x) = 0$, leżący w przedziale $[A,B]$ z żadaną dokładnością.

Podprogram ten wymaga spełnienia warunku:

$$F(A) \cdot F(B) < 0,$$

co oznacza, że funkcja $F()$ na końcach przedziału $[A,B]$ przyjmuje odmienne znaki. Obliczenie pierwiastka dokonuje się metodą siecznych /regula falsi/, której opis można znaleźć niemal w każdym podręczniku metod numerycznych.

Skala:

W obliczeniach występują jedynie liczby mniejsze od 2, zatem wystarczającą będzie skala 1.

Dokładność:

Zadaną dokładnością obliczeń jest uzyskanie czterech cyfr do-

kładnych po kropce pozycyjnej. Ponieważ, licząc w skali 1, maszyna liczy z dziewięcioma cyframi po przecinku, zatem żądana dokładność będzie łatwo osiągnięta.

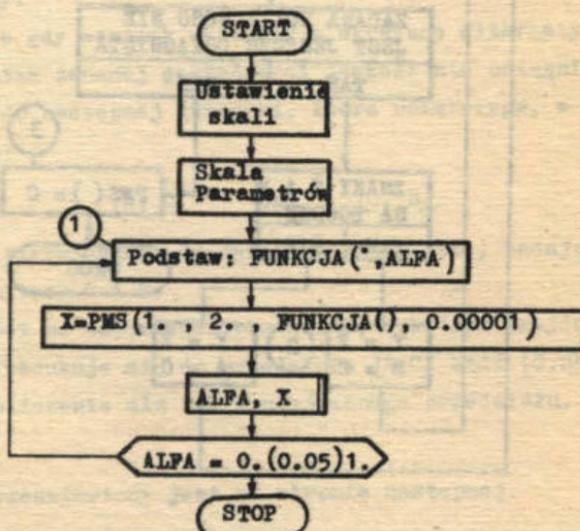
Postać danych:

Wszelkie parametry liczbowe występują bezpośrednio w programie, zatem danych wejściowych problemu w tym przypadku brak.

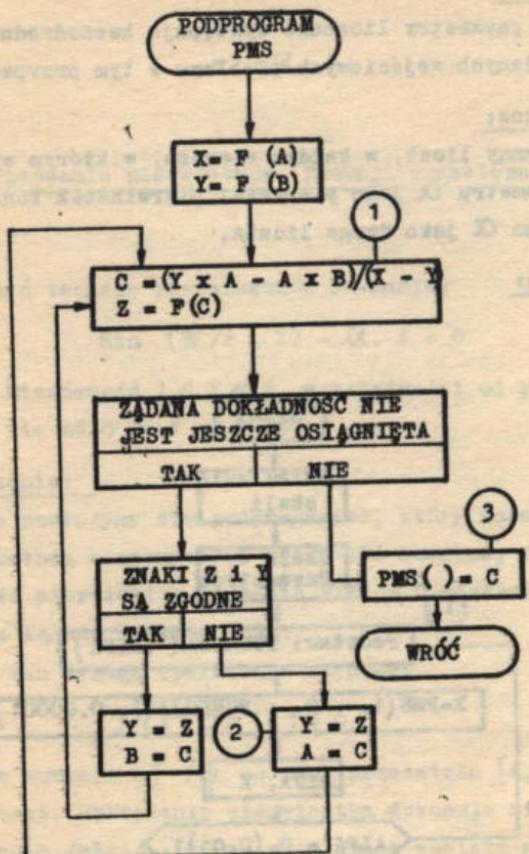
Postać wyników:

Dwie kolumny liczb, w każdym wierszu, w którym wypisana jest wartość parametru α jako pierwsza, pierwiastek funkcji odpowiadający danemu α jako druga liczba.

Sieć działań:



Sieć działań programu głównego



Rysunek 13
Sieć działań podprogramu PMS

Objaśnienia sieci działań:

Sieć działań programu głównego wymaga objaśnienia jedynie dla skrzynki, oznaczonej numerem 1. Powoduje ona podstawienie właściwej wartości parametru ALFA jako drugiego argumentu w funkcji, której pierwiastków szukamy. Wartość ta jest wyznaczona po przez skrzynkę POWTÓRZ.

Sieć działań podprogramu PMS realizuje następujące czynności:

1. Obliczenie przybliżenia C szukanego pierwiastka drogą zastąpienia łuku krzywej przez sieczną /pierwsze dwie skrzynki/.
2. Sprawdzenie, czy C nie przybliża pierwiastka z żądaną dokładnością /pierwsza skrzynka alternatywy/. W przypadku, gdy osiągnięta została żądana dokładność przybliżenia, następuje określenie wartości funkcji PMS /skrzynka opatrzona numerem 3/ i powrót do programu głównego. Drodze tej odpowiada "NIE" w skrzynce alternatywy.

W przypadku gdy warunek wypisany w skrzynce alternatywy jest spełniony, zatem żąданej dokładności jeszcze nie osiągnięto, przechodzimy do następnej skrzynki, która rozstrzyga, w którym z przedziałów:

$$[A, C] \quad \text{czy} \quad [C, B]$$

leży szukany pierwiastek. Odpowiedź uzyskuje się, badając znaki funkcji w punktach C i B.

w zależności od odpowiedzi przedział, w którym znajduje się pierwiastek, redukuje się do przedziału $[A, C]$ bądź $[C, B]$ i powtarza się obliczenie dla tak zmniejszonego przedziału.

Program:

Program przedstawiony jest na stronie następnej.

Objaśnienie programu:

Pierwsza część programu, tak zwany program główny, jest prostym rozpisaniem sieci działań. Należy zwrócić uwagę na podstawienie wielkości ALFA jako drugiego parametru FUNKCJA(), określonego odpowiednim podprogramem.

K) TABLICOWANIE PIERWIASTKOW FUNKCJI PRZESTEPNEJ

USTAW SKALE DZIESIETNIE: 1
SKALA DZIESIETNA PARAMETROW: 1

- 1) PODSTAW : FUNKCJA(., ALFA)
 $X = \text{PMS}(1., 2., \text{FUNKCJA}(), 0.00001)$
LINIA
DRUKUJ(10.5): ALFA, X
POWTORZ OD 1: ALFA = 0.(0.05)1.
STOP 1

PODPROGRAM: PMS(A, B, F(), DOKLADNOSC)

- $X = F(A)$
 $Y = F(B)$
1) $C = (Y \times A - X \times B) / (Y - X)$
 $Z = F(C)$
GDY ABS(Z) > DOKLADNOSC: NASTEPNY, INACZEJ 3
GDY SGN(1, Z) = SGN(1, Y): NASTEPNY, INACZEJ 2
 $Y = Z$
 $B = C$
SKOCZ DO 1
- 2) $X = Z$
 $A = C$
SKOCZ DO 1
- 3) $\text{PMS}() = C$
WRÓC

PODPROGRAM: FUNKCJA(X, ALFA)

$\text{FUNKCJA}() = \sin(1.570796 \times X) - ALFA \times X$
WRÓC

KONIEC

Kropka, występująca w miejscu pierwszego argumentu wskazuje na to, że argument ten w chwili wykonania zdania PODSTAW nie jest jeszcze określony. /Określają go dopiero zdania podprogramu PMS()./.

Podprogram PMS() jest podprogramem o czterech argumentach, z których trzeci nie przedstawia sobą liczby, lecz funkcję, której pierwiastek mamy znaleźć. W momencie wykonania zdania programu głównego, powodującego obliczenie wielkości X, symbol ogólny F() zostaje zastąpiony przez konkretną nazwę FUNKCJA() reprezentującą drugi podprogram, zdefiniowany w programie.

Zdanie GDY ABS(Z) > DOKŁADNOŚĆ realizuje pierwszą skrzynkę alternatywy. Druga skrzynka alternatywy bada zgodność znaków Z i Y przy pomocy funkcji SGN. Dalsza część zdania podprogramu PMS jest ścisłym powtórzeniem czynności, opisanych w sieci działań.

Drugim z podprogramów jest podprogram FUNKCJA(X, ALFA) którego można nie objaśniać ze względu na jego prostotę. Z podprogramu tego korzysta podprogram PMS /mamy więc do czynienia z dwustopniowym użyciem podprogramów/. Parametr ALFA jest podstawiany przez program główny.

Sprawdzenie programu:

Sprawdzenia można dokonać przez znajdywanie przy pomocy programu głównego i podprogramu PMS znanych pierwiastków pewnych funkcji.

Sprawdzenie wyników:

Wyniki sprawdzamy dla α równego 0 i 1, dla których to wartości powinniśmy otrzymać pierwiastki równe odpowiednio 2 i 1.

Wyniki:

+0.00000	+2.00000
+0.05000	+1.93821
+0.10000	+1.87962
+0.15000	+1.82367
+0.20000	+1.76960
+0.25000	+1.71747
+0.30000	+1.66666
+0.35000	+1.61701
+0.40000	+1.56250

+0.45000	+1.52038
+0.50000	+1.47296
+0.55000	+1.42603
+0.60000	+1.37937
+0.65000	+1.33288
+0.70000	+1.28641
+0.75000	+1.23983
+0.80000	+1.19299
+0.85000	+1.14575
+0.90000	+1.09797
+0.95000	+1.04945
+1.00000	+1.00000

Nowe pojęcia SAKO:

PODSTAW	s. 139
Korzystanie z podprogramów	s. 155

10. Tablicowanie funkcji SI(X)

Po dany przykład jest już znacznie bardziej rozbudowany niż po przednie i zakłada dobre zaznajomienie się z poprzednim.

Problem:

Należy ułożyć tablicę wartości funkcji SI X :

$$SI(X) = \int_0^X \frac{\sin(u)}{u} du$$

w zakresie, z krokiem i dokładnością określonymi przez dane wejściowe.

Metoda rozwiązywania:

Funkcję podcałkową $S(X) = \frac{\sin(X)}{X}$ rozwiniemy na szereg w pobliżu osobliwości $X = 0$ i zatrzymamy tylko pewną ilość wyrazów rozwinięcia, to znaczy przedstawimy je w postaci:

$$S(x) = \begin{cases} 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \frac{x^8}{9!} & \text{dla } x \leq 0.1 \\ \sin(x) / x & \text{dla } x > 0.1 \end{cases}$$

Odrzucenie pozostałych wyrazów rozwinięcia powoduje powstanie błędu, nie większego niż $\frac{x^{10}}{11!}$, co dla $x \leq 0.1$ daje liczbę mniejszą, niż 10^{-10} , co przekracza dokładność maszyny.

Do całkowania funkcji $S(x)$ użyjemy bibliotecznego podprogramu CALKA.

Skala:

Skala obliczeń podawana jest jako jedna z danych wejściowych.

Dokładność:

Dokładność obliczenia całki określona jest jako jedna z danych wejściowych. Podprogram CALKA jest określony tak, że zadana dokładność jest automatycznie osiągnięta /o ile nie przekracza ona dokładności maszyny/.

Postać danych:

Dane wejściowe stanowią cztery liczby, wypisane jedna pod drugą i opatrzone ewentualnie komentarzami. Wypisane są kolejno liczby, określające skalę obliczeń N, odstęp tablicowania KROK, graniczną wartość tablicy GRANICA oraz dokładność: DOKŁADNOŚĆ. W naszym konkretnym przypadku przyjmiemy następujące dane:

```

:=2
KROK=0.1
GRANICA=2.
DOKLADNOSC=.0001

```

Postać wyników:

Wyniki mogą być opatrzone nagłówkiem, stanowiącym tytuł tablicy, przy czym w nagłówku mają być wymienione dane wejściowe programu.

Tablica ma się składać z trzech kolumn, zaopatrzonych nagłówkami: NR, X, SI(X), pod odpowiednim nagłówkiem ma występować:

- numer kolejny pozycji tablicy, począwszy od 1,
- wartość X,
- odpowiadająca powyżej wartości SI(X).

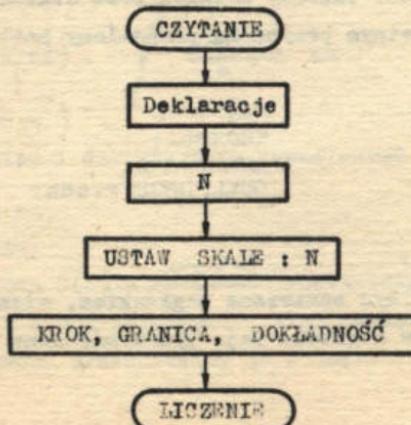
Sieć działań: na stronach 78, 79, 80, 81.

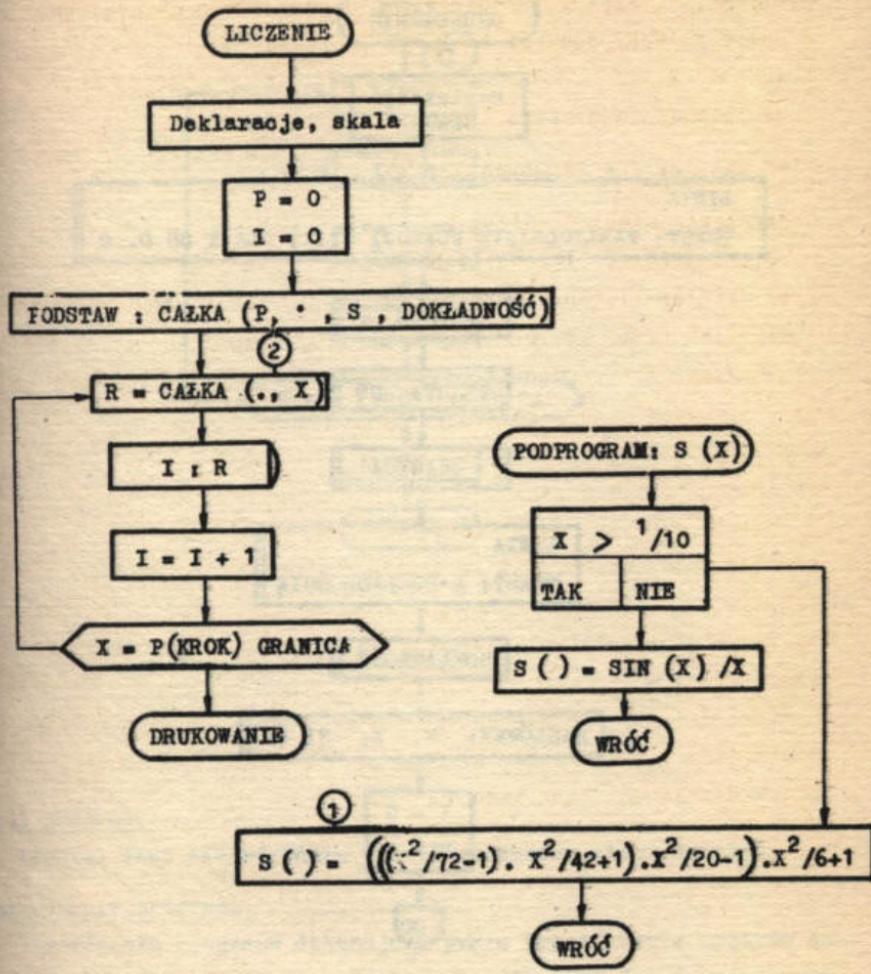
Objaśnienie sieci działań:

Sieci działań CZYTANIA i DRUKOWANIA nie wymagają komentarzy.
Sieć działań LICZENIE wskazuje na "wyniesienie" jednorazowych podstawień przed pętlą względem X, przy pomocy zdania PODSTAW. Dla każdego X odpowiednia całka zostaje zapisana w pamięci bębnowej, poczynając od zerowego miejsca tej pamięci. Rozdział 3 (DRUKOWANIE) pobiera te wartości z pamięci bębnowej, przekazując je do drukowania.

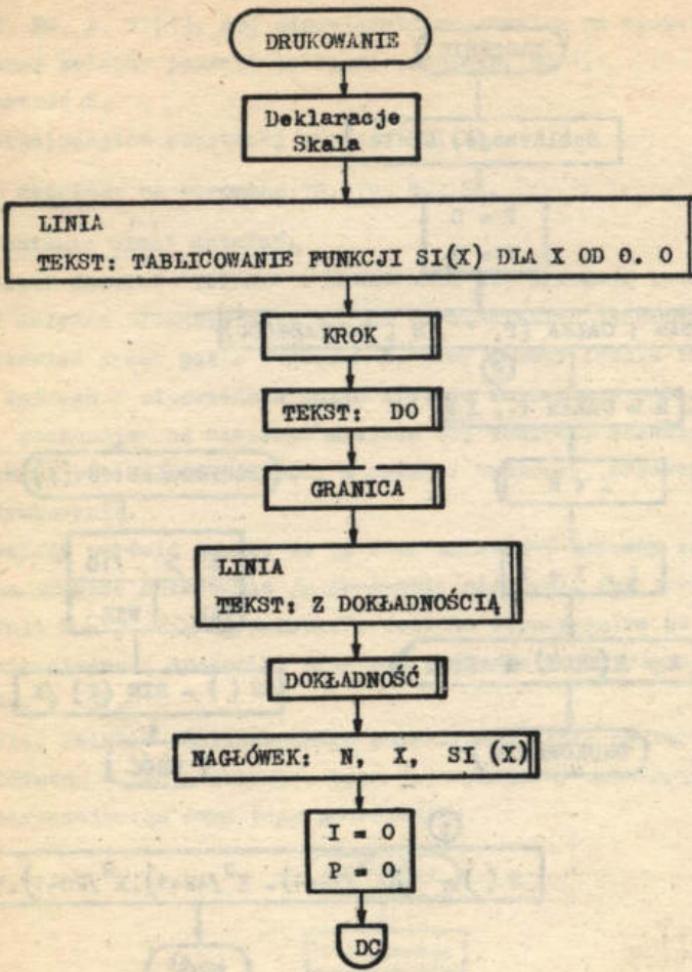
Należy zwrócić uwagę, że tę samą zmienną X używamy raz jako górną granicę całkowania /w programie głównym/, raz jako argument funkcji S (X) /w programie pod/. Jest to dopuszczalne ze względu na niezależność symboliki programu głównego i podprogramów /patrz rozdz. V, 3/.

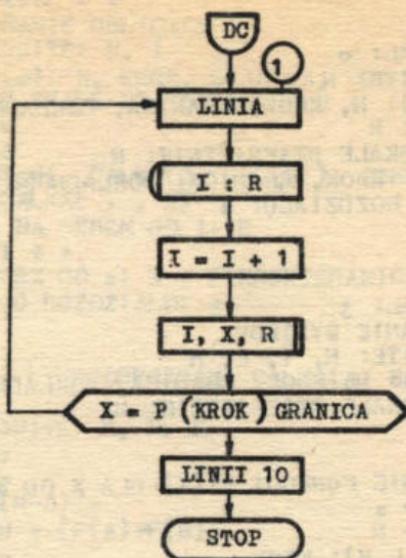
Sieć działań bibliotecznego podprogramu CAŁKA opuszczamy jako nieistotną w tym problemie. Opis jej wymagałby uzasadnienia, przekraczającego ramy tego podręcznika.





Rysunek 14





Opis programu:

Program jest bezpośrednim przetłumaczeniem sieci działań.

Sprawdzenie programu:

Sprawdzenia programu dokonujemy przez przeliczenie wyników dawanych dla różnych danych wejściowych, dla wartości X mniejszych i większych od 0.1.

Sprawdzenie wyników:

Sprawdzenia wyników dokonujemy poprzez dwukrotne obliczenie tablicy, po raz wtóry być może ze zwielokrotnionym krokiem, co pozwala nam sprawdzić tylko pewne wartości tablicy.

Postać wyników dla danych określonych w punkcie 5.

K) TABLICOWANIE FUNKCJI SI(X)

ROZDZIAŁ: 0
CALKOWITE: N
BLOK(o): N, KROK, GRANICA, DOKLADNOSC
CZYTAJ: N
USTAW SKALE DZIESIETNIE: N
CZYTAJ: KROK, GRANICA, DOKLADNOSC
IDZ DO ROZDZIAŁU: 2

ROZDZIAŁ: 3
K) DRUKOWANIE WYNIKOW
CALKOWITE: N, I, L, M
BLOK(o): N, KROK, GRANICA, DOKLADNOSC
USTAW SKALE DZIESIETNIE: N
LINIA
TEKST:
TABLICOWANIE FUNKCJI SI(X) DLA X OD 0 DO CO
L = N + 2
M = 9 - N
DRUKUJ(L..M): KROK
SPACJA
TEKST:
DO
DRUKUJ(L..M): GRANICA
LINIA
SPACJ 15
TEKST:
Z DOKLADNOSCIA
DRUKUJ(3..M): DOKLADNOSC
LINIT 4
L = 9 + N
TEKST WIERSZY i:
NR. X SI(X)
LINIT 2
I = 0
P = 0
*) LINIA
CZYTAJ Z BEDNA OD I: R
I = I + 1
DRUKUJ(10): I
DRUKUJ(L..H): X, R
POUTORZ OD i: X = P(KROK)GRANICA
LINIT 10
STOP NASTEPNY
IDZ DO ROZDZIAŁU: 0

ROZDZIAŁ : 2

K) WYKONANIE OBliczen
CALKOWITE: N, I
BLOK(o): N, KROK, GRANICA, DOKLADNOSC
USTAW SKALE DZIESIETNIE: N
P = 0
I = 0
PODSTAW: CALKA(P, . , S(), DOKLADNOSC, o)
• 2) R = CALKA(. , X)
PISZ NA DEBEN OD I: R
I = I + 1
POWTORZ OD 2: X = P(KROK)GRANICA
IDZ DO ROZDZIAŁU: 3

PODPROGRAM: CALKA(A, B, F(), EPSILON, N)

CALKOWITE: M, N, K
H = 1
Q = 0
H = (B-A)/2
J = H × (F(A)+F(B))
2) S = 0
• 2) S = S + H × F(A+H×(2×K-1))
POWTORZ OD 1: K = 1(1)H
I = J + 4×S
GDY N>M: 3, INACZEJ NASTEPNY
GDY EPSILON > ABS(I-Q): NASTEPNY, INACZEJ 3
CALKA() = I/3
WRÓC
3) Q = 1
J = (1+J)/4
M = M × 2
H = H / 2
SKOCZ DO 2

PODPROGRAM: S(X)

GDY X > 1/10 : NASTEPNY, INACZEJ 1
S() = SIN(X)/X
WRÓC
1) S() = (((X×2/72-1)×X×2/42+1)×X×2/20-1)×X×2/6+1
WRÓC

KONIEC

TABLICOWANIE FUNKCJI SI(X) DLA X OD 0 DO +2.0000000
Z DOKŁADNOŚCIĄ +0.0001000

NR	X	SI(X)
1	+0.0000000	+0.0000000
2	+0.1000000	+0.0999445
3	+0.2000000	+0.1995561
4	+0.3000000	+0.2985041
5	+0.4000000	+0.3964615
6	+0.5000000	+0.4931075
7	+0.6000000	+0.5881291
8	+0.7000000	+0.6812229
9	+0.8000000	+0.7720971
10	+0.9000000	+0.8604709
11	+1.0000000	+0.9460933
12	+1.1000000	+1.0286856
13	+1.2000000	+1.1080478
14	+1.3000000	+1.1839588
15	+1.4000000	+1.2562279
16	+1.5000000	+1.3246851
17	+1.6000000	+1.3891826
18	+1.7000000	+1.4495024
19	+1.8000000	+1.5058170
20	+1.9000000	+1.5577756
21	+2.0000000	+1.6054133

11. Rozwiązywanie równania różniczkowego

Problem:

Należy ułożyć program na obliczanie przybliżonych wartości funkcji $y(x)$, będącej rozwiązaniem równania różniczkowego:

$$/1/ \quad y'' + a \cdot y' + b \cdot y^3 + f^2 : y = \sin (F \cdot x)$$

dla warunków początkowych:

$$y(0) = 0$$

$$y'(0) = f$$

Wartości funkcji należy wyznaczyć w punktach $x = 0.0, 0.05, 0.10, \dots, 0.95, 1.00$ z dokładnością 0.001.

Dane są następujące wartości parametrów:

$$a = 0.35$$

$$b = 0.012$$

$$f = 6.5$$

$$F = 2.8$$

Metoda rozwiązań:

Przyjmujemy następujące funkcje pomocnicze:

$$y_0(x) = x$$

$$/2/ \quad y_1(x) = y(x)$$

$$y_2(x) = y'(x) / f$$

Rozwiązanie równania $/1/$ sprowadza się do znalezienia trzech funkcji $y_0(x), y_1(x), y_2(x)$. Układ równań określających te trzy funkcje, ma postać:

$$y'_0(x) = 1$$

$$/3/ \quad y'_1(x) = y'(x)$$

$$y'_2(x) = y''(x) / f$$

Na mocy $/1/$ mamy zależność:

$$y''(x)/f = [\sin(F \cdot x) - a(y'(x) + b \cdot y(x)^3) - f^2 \cdot y(x)] / f$$

i podstawiając z $/2/$

$$y(x) = f \cdot y_2(x)$$

$$y(x) = y_1(x)$$

otrzymujemy:

$$y''(x)/f = [\sin(F \cdot x) - a(f \cdot y_2(x) + b \cdot f^3 \cdot y_2(x)^3) - f^2 \cdot y_1(x)]/f$$

$$= \frac{1}{f} \sin(F \cdot x) - a(y_2(x) + b \cdot f^2 \cdot y_2(x)^3) - f \cdot y_1(x)$$

1 ostatecznie otrzymujemy układ:

$$y'_0 = 1$$

$$/4/ \quad y'_1 = f \cdot y_2$$

$$y'_2 = \sin(F \cdot x) /f - a(y_2 + b \cdot f^2 \cdot y_2^3) - f \cdot y_1$$

z warunkami początkowymi, otrzymanymi z /2/

$$y_0(0) = 0$$

$$y_1(0) = 0$$

$$y_2(0) = y'(0)/f = f/f = 1.$$

Postać /4/ nadaje się do zastosowania podprogramu RK /całkowanie układów równań różniczkowych metodą Runge Kutta/, znajdującego się w bibliotece podprogramów.

Skala:

Dla określonych wyżej parametrów, danych początkowych i zakresu obliczeń, żadna z wielkości, występujących w zadaniu, nie przekracza liczby 100. Przyjmijmy wobec tego skalę dziesiętną 2.

Dokładność:

Licząc w skali dziesiętnej 2 używamy ośmiu cyfr dziesiętnych po przecinku. Ponieważ żądana dokładność to trzy cyfry dokładne po przecinku, zatem dla tak określonego zadania żądana dokładność będzie osiągnięta. Patrz jeszcze punkt "Sprawdzanie wyników".

Postać danych wejściowych:

Dane wejściowe w tym zadaniu wprowadzamy wraz z programem.

Postać wyników:

Wyniki winny być przedstawione w postaci następującej tablicy:

ROZWIĄZANIA RÓWNANIA RÓŻNICZKOWEGO

PROBLEM 154/63, 10.11.1963

X

Y

d. dd	dd. ddd
d. dd	dd. ddd
d. dd	dd. ddd
.	.
.	.
.	.
d. dd	dd. ddd

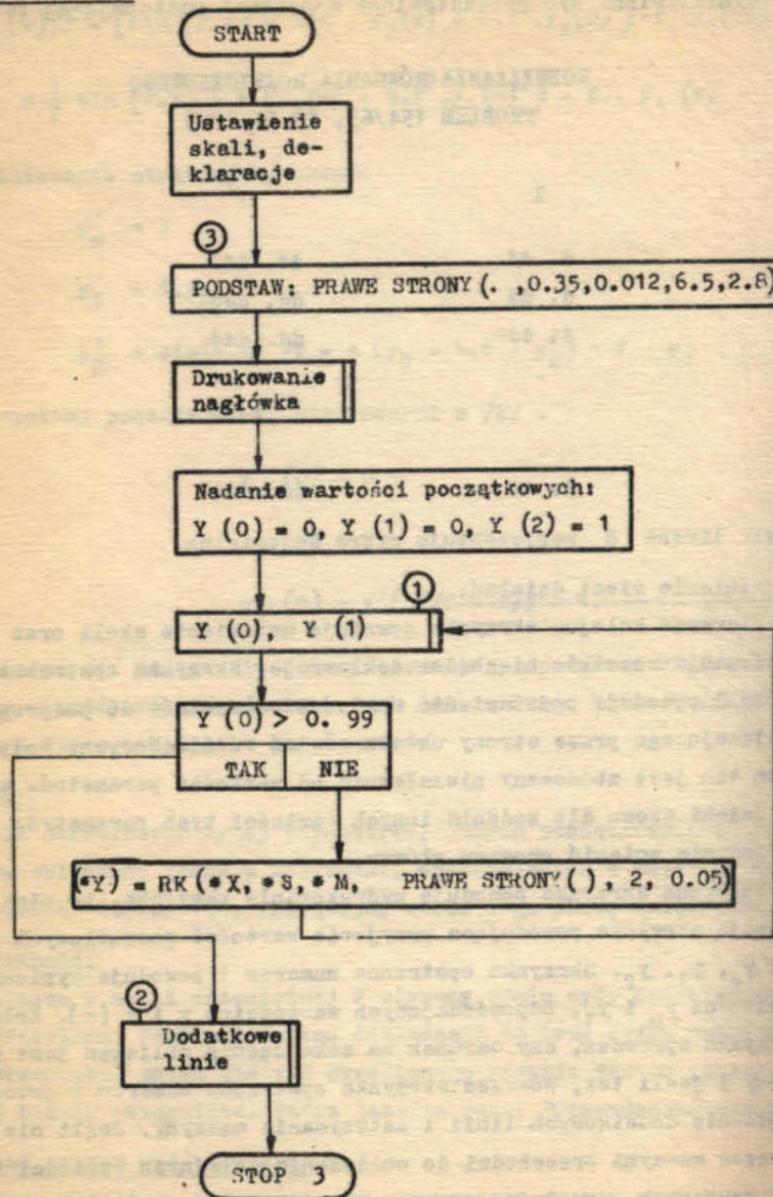
gdzie litera d reprezentuje cyfrę dziesiętną.

Objaśnienie sieci działań:

Pierwsza kolejna skrzynka powoduje ustawienie skali oraz reprezentuje wszelkie niezbędne deklaracje. Skrzynka opatrzona numerem 3 powoduje podstawienie wartości parametrów do podprogramu obliczającego prawe strony układu równań różniczkowych. Podprogram ten jest zbudowany niezależnie od wartości parametrów a,b,f, F, dzięki czemu dla zadania innych wartości tych parametrów należy jedynie zmienić program główny.

Następna skrzynka powoduje wydrukowanie nagłówka, po niej następuje skrzynka powodująca przyjęcie wartości początkowych funkcji y_0 , y_1 , y_2 . Skrzynka opatrzona numerem 1 powoduje wypisanie wielkości y_0 i y_1 , odpowiadających wartościom x i y (x). Kolejna skrzynka sprawdza, czy warunek na zakończenie obliczeń jest spełniony i jeśli tak, wówczas skrzynka opatrzona numerem 2 powoduje wypisanie dodatkowych linii i zatrzymanie maszyny. Jeśli nie, wówczas maszyna przechodzi do obliczenia kolejnych wartości funkcji szukanych, wywołując program RK i przechodząc do skrzynki opatrzonej numerem 1.

Sieć działań



K) ROZWIĄZANIE RÓWNANIA ROZNICZKOWEGO

SKALA DZIESIETNA PARAMETROW: 2
USTAW SKALE DZIESIETNIE: 2
BLOK(a):: X, S, M, Y

3) PODSTAW: PRAWE STRONY(., 0.35, 0.012, 6.5, 2.8)
LINII 15

TEKST WIERSZY 6:

ROZWIĄZANIE RÓWNANIA ROZNICZKOWEGO
PROBLEM 154/63, 10.II.1963

X

Y

$$\begin{aligned} Y(0) &= 0 \\ Y(1) &= 0 \\ Y(2) &= 1 \end{aligned}$$

1) LINIA

DRUKUJ(12.2): Y(0)

DRUKUJ(14.3): Y(1)

GDY Y(0) > 0.99: 2, INACZEJ NASTĘPNY

(*Y) = RK'*X, *S, *M, PRAWE ST ONLY(), 2, 0.05)

SKOCZ DO 1

2) LINII 10

STOP 3

PODPROGRAM: (*M) = PRAWE STRONY(*X, A, B, FEMALE, FDUZF)

STRUKTURA(a): M, X

$$M(0) = 1$$

$$M(1) = \text{FEMALE} \times X(0)$$

$$M(2) = \text{SIN}(FDUZF \times X(0)) / (\text{FEMALE} - A \times (X(0) + B \times \text{FEMALE} \times X(0) \times 3) - \text{FEMALE} \times X(0))$$

WROC

PODPROGRAM: (*Y) = RK(*X, *S, *M, F(), N, H)

CALKOWITE: N, *A, I, K
STRUKTURA(N) : X, Y, S, M
TABLICA(3): A

I 2 2 I

*

*1) M(I) = 0

S(I) = 0

POWTORZ OD 1: I = 0(1)N

**2) X(I) = Y(I) + HxM(I)/A(K)

POWTORZ OD 2: I = 0(1)N

(*M) = F(*X)

*3) S(I) = S(I) + f(I)x(I)x(A(K))

POWTORZ OD 3: I = 0(1)N

POWTORZ OD 2: K = 0(1)3

*4) Y(I) = Y(I) + S(I)/6

POWTORZ OD 4: I = 0(1)N

WROC

KONIEC

Objaśnienie programu:

Zadeklarowanie czterech bloków X, Y, M, S o wymiarach 2 jest konieczne dla stosowania programu RK. Bloki te są przeznaczone dla przechowywania następujących wielkości:

blok X - program RK zapisuje w tym bloku argumenty funkcji PRAWE STRONY;

blok Y - program RK zapisuje w nim kolejny układ rozwiązań danego układu równań różniczkowych. Na początku obliczeń program główny zapisuje w tym bloku wartości początkowe rozwiązań;

blok M - program PRAWE STRONY zapisuje w tym bloku wartości prawych stron równań różniczkowych, obliczone dla argumentów, podanych w bloku X;

blok S - blok pomocniczy dla programu RK.

Obliczenie kolejnego układu rozwiązań odbywa się przy pomocy formuły

$$(*Y) = \text{RK}(*X, *S, *M, \text{PRAWE STRONY}(), 2, 0.05).$$

Znaczenie argumentów Y, X, S i M zostało podane powyżej. Argument PRAWE STRONY() jest nazwą podprogramu, obliczającego prawe strony równań. Liczba 2 jest wymiarem bloków X, Y, S, M, liczba 0.05 - krokiem całkowania.

Powysze argumenty są podstawiane w miejsce argumentów formalnych podprogramu RK, które występują w deklaracji podprogramu:

$$(*Y) = \text{RK}(*X, *S, *M, F(), N, H).$$

Z wyjątkiem parametru N wszystkie podstawiane wielkości powinny być ułamkowe.

Do programu dołączony jest podprogram obliczania prawych stron, PRAWE STRONY, którego argumentami są:

blok M - znaczenie tego bloku jest podane wyżej, tu podprogram zapisuje wyniki;

blok X - w tym bloku podprogram znajduje argumenty

A, B, FEMALE, FDUZE - oznaczają parametry pomocnicze dla obliczania prawych stron równań. Dzięki przyjęciu parametrów jako argumentów formalnych, podprogram PRAWE STRONY może być stosowany dla różnych układów wartości parametrów.

Ponadto do programu dołączony jest standartowy podprogram RK, którego wyjaśnienie wykracza poza ramy tego podręcznika.

Sprawdzanie programu:

Program sprawdzamy dla uproszczonego podprogramu PRAWE STRONY dającego łatwe do sprawdzenia wyniki na przykład dla podprogramu realizującego następujący układ zależności:

$$h_0(y_0, y_1, y_2) = 1$$

$$h_1(y_0, y_1, y_2) = F - f + a - b$$

$$h_2(y_0, y_1, y_2) = 0$$

Podprogram PRAWE STRONY sprawdzamy dla kilku układów wartości $x(0)$, $x(1)$, $x(2)$. Stosujemy w tym celu pewien prosty program główny.

Podprogram RK, jako biblioteczny, nie wymaga sprawdzania.

Sprawdzanie wyników:

Wyniki sprawdzamy przez dwukrotne liczenie programu. Ponizej przedstawione są wyniki programu:

ROZWIĄZANIE RÓWNANIA ROZNICZKOWEGO
PROBLEM 154/63, 10.11.1963

X	Y
+0.00	+0.000
+0.05	+0.315
+0.10	+0.590
+0.15	+0.799
+0.20	+0.922
+0.25	+0.948
+0.30	+0.877
+0.35	+0.718
+0.40	+0.490
+0.45	+0.219
+0.50	-0.066
+0.55	-0.335
+0.60	-0.561
+0.65	-0.722
+0.70	-0.802
+0.75	-0.795
+0.80	-0.704
+0.85	-0.540
+0.90	-0.321
+0.95	-0.073
+1.00	+0.179

Sprawdzamy ponadto dokładność otrzymanych wyników następującą metodą. Jak wiadomo z analizy numerycznej, przybliżone rozwiązania równania różniczkowego otrzymane metodą Runge-Kutta podaną w przykładzie różnią się od rozwiązań dokładnych o wielkość

$$K \cdot h^5,$$

gdzie h w naszym przypadku jest równe 0.05, a czynnik K zależy od postaci równania, zakresu obliczeń oraz kroku całkowania h . Przyjmuje się, iż dla różnych h wielkość K jest w przybliżeniu stała

$$K \approx \text{constans}$$

Licząc zatem problem drukrotnie, raz dla $h = 0.05$, drugi raz dla $h_1 = h/2$, otrzymujemy w każdym interesującym nas ustalonym punkcie x przedziału dwa rozwiązania: $y(x)$ i $y_1(x)$. Jeśli przez $y_0(x)$ oznaczymy dokładne rozwiązanie równania różniczkowego w punkcie x , otrzymujemy następujący układ zależności:

$$y(x) - y_0(x) \approx K \cdot h^5$$

$$y_1(x) - y_0(x) \approx K \cdot h_1^5 = h(4/2)^5 = \frac{K}{32} \cdot h^5$$

wyznaczając z tego układu $K \cdot h^5$ otrzymujemy:

$$K \cdot h^5 \approx \frac{y(x) - y_1(x)}{\left(1 - \frac{1}{32}\right)} = \frac{32}{31} (y(x) - y_1(x))$$

i ostatnia otrzymana wielkość, znana na podstawie dwukrotnego rozwiązania problemu, stanowi pewną miarę uzyskanej dokładności liczenia.

ROZDZIAŁ 3

FORMA I OPIS POJĘĆ PODSTAWOWYCH W JĘZYKU SAKO

Poniżej podany jest opis i forma tych wyrażeń, które będą wykorzystane w opisie rozkazów SAKO /rozdz. 4/ i które posiadają jednakowe znaczenie, niezależnie od rozkazu, w którym będą one używane.

1. Liczby

a. L i c z b a u ła m k o w a

Przykłady:

45.678

007

2345

2.716281828

123456789.1

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 między kolejnymi cyframi.

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 biorąc, liczba zapisana w skali dziesiętnej N składając się może z N cyfr przed kropką pozycyjną oraz z $10-N$ cyfr po kropce. Ścisłe 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	$a_N = 2^k$	K
0	1	0
1	16	4
2	128	7
3	1024	10
4	16384	14
5	131072	17
6	1 048576	20
7	16 777216	24
8	134 217728	27
9	1073 741824	30
10	34359 748368	35

W powyższej tabeli a_N przedstawia najmniejszą liczbę o postaci 2^k , która jest większa od 10^N . Wyjątkiem jest tutaj a_{10} , które reprezentuje największą liczbę, którą można zapisać w pamięci maszyny, zwiększoną o 1.

Skalę dziesiętną dla liczb ułamkowych, występujących explicite w programie, podaje deklaracja SKALA DZIESIĘTNA PARAMETRÓW. 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 DZIESIĘTNIE 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 błędnym 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 BYŁ NADMIAR. Wykonanie tego rozkazu powoduje m. in. wyzerowanie wskaźnika nadmiaru.

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

Przykłady:

321
7
45678
15
7654

Znaczenie:

Ogólnie przyjęte w matematyce.

Forma:

Ciąg co najwyżej pięciu cyfr dziesiętnych.

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 mieszcząca się w komórce pamięci /jest nią każda liczba, większa od 131 071/, powstaje nadmiar z wszystkimi jego konsekwencjami.

c. 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 algebry 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.

d. 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 /po-
czawszy od zera/ liczb naturalnych, przebieganych przez pewien
indeks, nosi nazwę zakresu tego indeksu.

Przykłady:

1. Penumerowana 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

$$a_{0,0} \ a_{0,1} \ a_{0,2} \ a_{0,3}$$

$$a_{1,0} \ a_{1,1} \ a_{1,2} \ a_{1,3}$$

$$a_{2,0} \ a_{2,1} \ a_{2,2} \ a_{2,3}$$

stanowi dwuwymiarowy blok, zwany macierzą 3x4. Każdy element macierzy jest opatrzony dwoma indeksami, których zakresy w tym przypadku wynoszą 3 i 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 przedstawia się następująco. 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 maszyny jako k-ty z kolei, 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 wyznaczony na przykład przez indeksy 1, 2, 2, 4

bedzie wczytany do pamięci maszyny jako

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

z kolei (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 tak zwane "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 nazw zmiennych prostych.

2. Zmienne

a. Zmienna prosta

Przykłady:

A

A12

EPSILON

R MALE

X3

SUMA KWADRATÓW

Znaczenie:

Jest to symbol przyjmujący w trakcie obliczeń różne wartości liczbowe.

Ta sama zmienna użyta w różnych rozdziałach posiada całkowicie niezależne znaczenie. 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 którą 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.

Na przykład 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/S } litery i cyfry

SDF } ciąg rozpoczyna się od cyfry
3H8 }

b. Z m i e n n a i n d e k s o w a n a

Przykłady:

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

Znaczenie:

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

Forma:

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

Reguły:

1. Posługiwanie 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 być tylko:

- liczby całkowite nieujemne,
- zmienne proste lub indeksowane o wartościach całkowitych nieujemnych,
- wyrażenia arytmetyczne o wartościach całkowitych nieujemnych /patrz wyrażenia arytmetyczne s. 104/.

3. Numery

Przykłady:

1
1 WYJŚCI.
2BC

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 podprogramu lub tego samego rozdziału.

Forma:

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

4. FunkcjePrzykłady:

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 definowane /patrz. s. 158/.

W SAKO wyróżniamy funkcje języka przedstawione tablicy 2. Inne funkcje jako definicje wprowadza się rozkazem PROGRAM.

Tablica 2

Skrót	Nazwa	
SIN(X)	Siinus	
COS(X)	Cosinus	
TG(X)	Tangens	
ASN(X)	Arcus sinus	
ACS(X)	Arcus cosinus	
ATG(X)	Arcus tangens	
ARC(X,Y)	Arcus	Oznacza kąt, leżący w óciartce 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
MOD(X,Y)	Modulo	Oznacza resztę powstałą z dzielenia X przez Y /tylko dla całkowitych X i Y/
SGN(X,Y)	Signum	Oznacza liczbę $ X \cdot \operatorname{sgn} Y$
ABS(X)	Wartość bezwzględna /absolutna/	Oznacza wartość absolutną X
ENT(X)	Część całkowita	Oznacza część całkowitą liczby X, w sensie SAKO
DIV(X,Y)	Dzielenie całkowite	Oznacza część całkowitą ilorazu X/Y /tylko dla całkowitych X i Y/
SKL(•)	Skala	Oznacza liczbę całkowitą, będącą aktualnie przyjętą skalą binarną

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 znaki, nie licząc spacji. Argumenty funkcji definiowanych podaje się w sposób identyczny, jak dla funkcji języka SAKO.

Uwagi:

1. Trzy pierwsze znaki nazwy funkcji definiowanej nie mogą pokrywać się z trzema pierwszymi literami funkcji języka.
2. Osiem ostatnich funkcji, podanych w tablicy 2 można używać jedynie dla oznaczenia wartości funkcji. Na przykład:

A = CAŁKA(A,B,ABS(),EPSILON) źle

A = CAŁKA(A,B+ABS(c),SIN(),0.01) dobrze

5. Wyrażenia arytmetyczne

Przykłady:

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 + - x / posiadają konwencjonalne znaczenie dodawania, odejmowania, mnożenia i dzielenia. Znak * jest symbolem potęgowania:

$A * B$ oznacza A^B .

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

Każde wyrażenie arytmetyczne ma jednoznacznie określona wartość powstałą 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:

1. Zmienna bądź liczba jest wyrażeniem.
2. Jeśli A jest wyrażeniem, wówczas (A) jest również wyrażeniem.
3. Jeśli A jest wyrażeniem nie rozpoczynającym się od znaku + lub -, wówczas $-A$, $+A$ jest również wyrażeniem.
4. Jeśli A, B są wyrażeniami, B nie rozpoczyna się od znaku + lub -, wówczas

$$A + B$$

$$A - B$$

$$A \times B$$

$$A / B$$

$$A * B$$

są również wyrażeniami.

5. Jeśli G jest nazwą bloku, A, B, ..., Z są wyrażeniami przyjmującymi wartości całkowite i ilości 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.

Powyższa definicja formy wyrażenia pozwala budować wyrażenia drogą składania ich z wyrażeń mniej skomplikowanych, bądź też

pezwala sprawdzać poprawność budowy konkretnych wyrażeń arytmetycznych.

Nie są wyrażeniami:

A + / B

A + -4.5678

A * * C

(A + B) C.

Kolejność wykonywania działań w wyrażeniu arytmetycznym.

W SAKO przyjmujemy następującą listę mocy działań:

1. Obliczanie wartości funkcji i wartości zmiennych indeksowanych.

2. Wykonywanie działań * /potęgowanie/.

3. Wykonywanie działań x.

4. Wykonywanie działań /.

5. Wykonywanie działań -.

6. Wykonywanie działań +.

Działaniem o większej mocy nazywać będziemy działanie, stojące na podanej liście pod mniejszym numerem.

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

Z dwu działań wykonuje się najpierw te, które jest zamknięte w większej ilości nawiasów. W przypadku równej ilości nawiasów wykonuje się najpierw działanie o większej mocy. Działanie o równej mocy zamknięte w jednakowej ilości nawiasów wykonuje się kolejne 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 lub 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 zdefiniowane 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.

5. Wyrażenie $A * B$ jest całkowite wtedy i tylko wtedy, gdy zarówno A i B są całkowite; w przypadku tym wyrażenie to ma sens tylko dla $B \geq 0$. W przypadku, gdy $A = 0$ i $B = 0$ maszyna sygnalizuje błąd; podobnie, gdy B jest ułamkowe i $A < 0$, maszyna sygnalizuje błąd.

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 DZIESIĘTNIE /BINARNIE/, bądź ostatnio napisaną deklaracją SKALA DZIESIĘTNIA /BINARNA/ PARAMETRÓW.

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 . 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 WSKAŹNIK NADMIARU i może być wykryte w programie za pośrednictwem rozkazu sterującego GDY BYŁ NADMIAR.

6. Wyrażenia bulowskie

Przykłady:

A + BETA

A + (-C) \times D28

C * (-2) + KSI

Gx000.007.777.760

A \times (-B) + (-A) \times B

Znaczenie:

Wyrażeniem bulowskim jest ciąg znaków, określający rodzaj i kolejność wykonywania działań bulowskich na słowach, których symbole występują w wyrażeniu. Każde wyrażenie bulowskie ma jednocześnie określona 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. Jeśli A jest wyrażeniem bulowskim, nie rozpoczynającym się od znaku -, wówczas -A jest wyrażeniem bulowskim.

3. Jeśli A jest wyrażeniem bulowskim, wówczas

(A)

jest również wyrażeniem bulowskim.

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

A + B

A x B

są wyrażeniami bulowskimi.

5. Jeśli A jest wyrażeniem, I - zmienną całkowitą lub liczbą 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 x,
3. wykonywanie sum i negacji + -.

Z dwu działań wykonuje się najpierw te, które jest zamknięte w większej ilości nawiasów. W przypadku równej ilości nawiasów wykonuje się te, które jest wymienione w liście hierarchii we wczesniejszej 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.

3. Wyrażenia bulowskie mogą występować jedynie w formułach bulowskich /patrz strona 137/, których forma różni się od formuł arytmetycznych zastąpieniem znaku = przez \equiv .

7. Lista

Przykłady:

A, B, C, D, E,
1, 2, 3, 4, 5, 6, 7
AX4, BT, *ALFA, *GAMMA,
1 WYJSCIE, 2CA, 8ICTB, -
3BX, 8
A(K+1), A(K+2), A(K+3), A(K+4)
A, B, F(), GXS(), #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, poprzedzone są symbolem *, na przykład:

* 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, to znaczy gdy z góry wiadomo, jakiego charakteru symbole występują na liście. Na przykład w deklaracji STRUKTURA, na liście na 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

lub

oz-

ż

mi,

sa

,

czy

is-

ej

h

za

po

Rozdział 4

FORMA I OPIS POSZCZEGÓLNYCH ROZKAZÓW SAKO

1. Deklaracje

ROZDZIAŁ: n

Przykłady:

ROZDZIAŁ: 0

ROZDZIAŁ: 12

ROZDZIAŁ: 348

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 albo KONIEC lub KONIEC ROZDZIAŁU jako rozdział i jednocześnie nadaje mu nazwę.

Forma:

Nazwą jest liczba naturalna, co najwyżej pięciocyfrowa.

Reguły:

1. Przed pierwszym wypisanym rozdziałem nie potrzeba dawać deklaracji ROZDZIAŁ, o ile nie jest potrzebny powrót do tego rozdziału /rozkaz sterujący IDŹ DO ROZDZIAŁU/. W szczególności deklaracja ROZDZIAŁ jest zbędna w programach, mieszących 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 danym rozdziale muszą być nadane wartości w tym samym rozdziale. Podobne 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 lub rozdziału o nazwie "n", jeśli program kończy się deklaracją "KONIEC: n". 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 : CAŁKA (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:

zmienne,
nazwy bloków,
nazwy podprogramów,
nazwy funkcji.

Jako wyniki występuować mogą:

zmienne,
nazwy 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, na przykład:

TRY ()

Reguła:

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

Uwagi:

1. W pamięci maszyny miejsce 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, DIV, SKL, ENT/ jak i funkcje określone przez podprogramy.

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.

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

Przykłady:

BLOK (15) : A, B

BLOK (3, 8) : 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.

Forma:

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. Zmienne indeksowane i deklaracje STRUKTURA muszą być przedzone w programie odpowiadającą im deklaracją BLOK.

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

Reguła ta staje się 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 : 0

BLOK (99) : B, C

CZYTAJ : * B

c (j) = -----

Nadawanie wartości zmiennym indeksowanym określającym elementy bloków B i C

ROZDZIAŁ : 1

BLOK (99) : D, E

W rozdziale 0 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) = . . .

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

W rozdziale 1 deklaracja BLOK powoduje również zarezerwowanie 200 miejsc w pamięci maszyny, określając jednocześnie strukturę bloków D i E. Jeżeli teraz, po wykonaniu rozdziału 0 rozkazem IDŹ DO ROZDZIAŁU : 1 przejdziemy do rozdziału 1, to konco-

we wartością bloków B i C rozdziału 0 stana się początkowymi wartościami bloków D i E rozdziału 1.

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

Przykłady:

STRUKTURA (ALFA) : A, B

STRUKTURA (N, 5) : BETA, G3

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

Znaczenie:

Deklaracja ta zmienia strukturę bloków, wymienionych w liście, a zadeklarowanych napisaną we wcześniejszej kolejności deklaracji 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 wspólnie 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 (1, 2) : A

3.7182

45.13

.1508

-.35

9.83

32.01

TABLICA (6) : 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.

Począnając 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

Nazwa 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 wprowadzania programu. Jest to istotna różnica w stosunku do deklaracji BLOK i STRUKTURA. Blokiem, określonym tymi deklaracjami, wartości nadawane są dopiero w trakcie wykonywania programu.

2. Gdy tablica składa się z liczb ułamkowych, skala, w jakiej mają być wprowadzone jej elementy, musi być ustalona za pośred-

nictwem 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

7321.0012.3754 111.000.111.000 721641002731

*

Znaczenie:

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

Forma:

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

Uwaga:

Patrz 1 i 2 uwaga dla deklaracji TABLICA.

CAŁKOWITE : lista

Przykłady:

CAŁKOWITE : A, GAMMA, X3

CAŁKOWITE : 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 CAŁKOWITE jest listą zmiennych, nazw bloków i nazw podprogramów. Słowo CAŁKOWITE oddzielone jest od pierwszego symbolu na liście dwukropkiem.

Reguły:

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

2. Deklaracja CAŁKOWITE 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 DZIESIĘTNA PARAMETRÓW: n

Przykłady:

SKALA DZIESIĘTNA PARAMETRÓW: 0

SKALA DZIESIĘTNA PARAMETRÓW: 3

Znaczenie:

Deklaracja ta ustala skalę dziesiętną, w jakiej będą wprowadzane /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,
 - w rozkazach POWTÓRZ.

SKALA BINARNA PARAMETRÓW : n

Przykłady:

SKALA BINARNA PARAMETRÓW : 0

SKALA BINARNA PARAMETRÓW : 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 DZIESIĘTNA PARAMETRÓW.

JEZYK SAS

Znaczenie:

Deklaracja ta stwierdza, że następujące po niej rozkazy są

napisane w języku SAS⁶. Powrót do języka SAKO wymaga deklaracji JEZYK SAKO.

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 : n

Znaczenie:

Deklaracja ta określa koniec napisanego programu. Liczba całkowita n oznacza nazwę rozdziału, od którego należy rozpoczęć wykonywanie programu.

Forma:

n - liczba całkowita.

Jeśli rozpoczęcie programu ma nastąpić od pierwszego napisanego rozkazu, wówczas znaki ": n" można opuścić.

Uwagi:

Po wprowadzeniu deklaracji KONIEC maszyna kończy wprowadzanie programu i w zależności od położenia kluczy na pulpicie sterującym maszyny lub od wypisanych dyrektyw operacyjnych /patrz rozdział VIII/ przechodzi do wypełnienia żądań programisty odnośnie danego rozdziału.

Mogą nimi być w szczególności:

- żądanie wykonania wprowadzonego programu
- żądanie wypisania wprowadzonego rozdziału w systemie oktalnym.

⁶ MASZYNA XYZ - Podręcznik programowania w języku maszyny. Systemy SAB, SAS i SO, Prace ZAM 1961 Ser. C Nr 1.
MASZYNA ZAM-2 - Kompendium programowania w języku SAS, Prace ZAM, 1962, Ser. C, Nr 2.

KONIEC ROZDZIAŁU

Znaczenie:

Deklaracja ta określa koniec napisanego rozdziału. Stosuje się ją głównie w przypadku tłumaczenia rozdziałów niezależnie od całego programu. Zazwyczaj po deklaracji KONIEC ROZDZIAŁU wypisuje się dyrektywy operacyjne, powodujące wypisanie tego rozdziału w systemie oktalnym oraz powodujące wykonanie pewnych innych czynności, opisanych w rozdziale VIII.

2. Rozkazy sterujące

SKOCZ 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 skróto NASTEPNY; rozkaz SKOCZ powoduje wówczas przejście do rozkazu wypisanego w najbliższej kolejności /staje się rozkazem nieefektywnym/.

SKOCZ WEDŁUG I : $\alpha, \beta, \dots, \omega$

Przykłady:

SKOCZ WEDŁUG ALFA : 1, 2A, NASTEPNY

SKOCZ WEDŁUG K2 : 3AB, 4, 2, 5C

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 NASTEPNY.

Uwaga:

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

POWTORZ OD α : I = J(K)L

Przykłady:

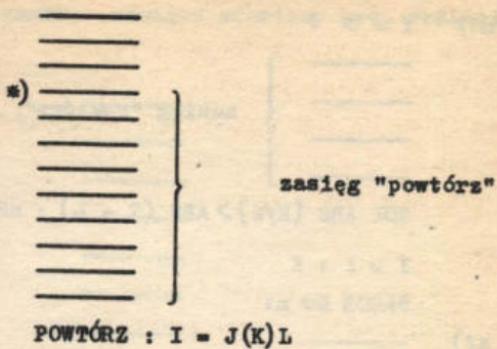
POWTORZ OD 3 : ALFA = 0(1)29

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

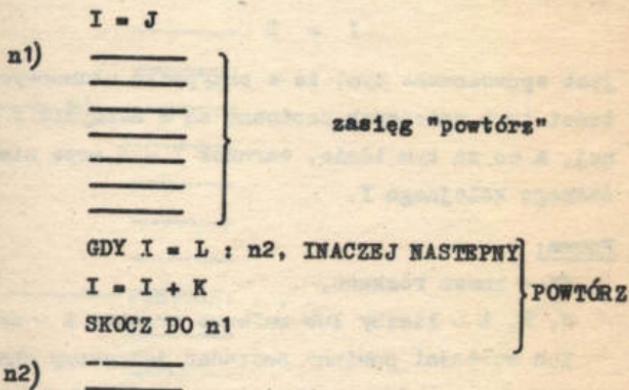
POWTORZ : AB1 = X0 (DELTA) 5

Znaczenie:

Rozkaz POWTORZ OD α : I = J(K)L powoduje wielokrotne wykonanie odcinka programu, zwanego zasięgiem danego rozkazu POWTÓRZ, a który jest zawarty między rozkazem POWTÓRZ a odpowiadającym mu znakiem *. Rozkaz POWTÓRZ powoduje wielokrotne wykonanie swojego zasięgu dla zmieniających się wartości zmiennej I. Pierwszy raz dany zasięg wykonuje się dla I = J, przy każdym następnym wykonaniu wartość I zwiększa się o K. Gdy zmienna I osiągnie wartość L, wówczas wykonuje się przejście do następnego rozkazu po rozkazie POWTÓRZ. Jeśli I, J, K, L - całkowite, wówczas odcinek programu:



jest równoważny następującemu:



gdzie n1, n2 są pewnymi numerami, różnymi od wszystkich używanych w danym rozdziale lub podprogramie.

Jeśli I, J, K, L - ułamkowe, wówczas równoważnym odcinkiem, przy poprzednich oznaczeniach, jest następujący:

n1) $I = J$

} zasięg "powtórz"

GDY $\text{ABS}(K/2) > \text{ABS}(I - L)$: n2, INACZEJ NASTĘPNY

$I = I + K$

SKOCZ DO n1

n2)

Ponieważ kolejne wartości, jakie przyjmuje zmieniona I, różnią się od siebie o wielkość K, zatem warunek, występujący w rozkazie GDY podanego przykładu można wysłowić następująco:

aktualna wartość I jest najbliższą wartości L spośród wszystkich możliwych wartości I.

Przyjęcie takiego warunku zamiast prostszego

$I = L$

jest spowodowane tym, że w przypadku ułamkowych I, j, K, L wartości tych zmiennych zapisane są w maszynie w postaci przybliżonej, a co za tym idzie, warunek $I = L$ może nie być spełniony dla żadnego kolejnego I.

Forma:

α - numer rozkazu,

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

Ich wartości powinny posiadać jednakowy charakter, tzn. winny być jednocześnie całkowitymi lub ułamkowymi. Słowa "OD α " nie posiadają istotnego znaczenia i mogą być opuszczone. Stosowanie ich jednak czyni program bardziej czytelnym.

Reguły:

1. Stosując rozkaz POWTÓRZ OD α stawiamy znak * z lewej strony numeru α ; jeśli stosujemy rozkaz POWTÓRZ od rozkazu,

nie zaopatrzonego w numer, wówczas stawiamy przy rozkazie tym znaki * :

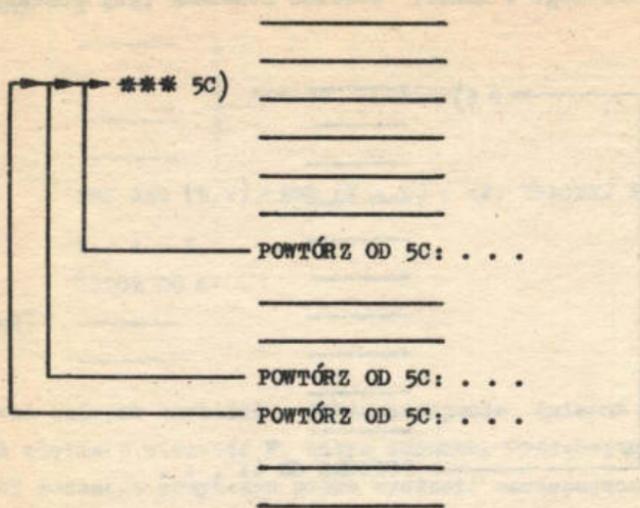
► * 6)

POWTÓRZ OD 6: . . .

► *)

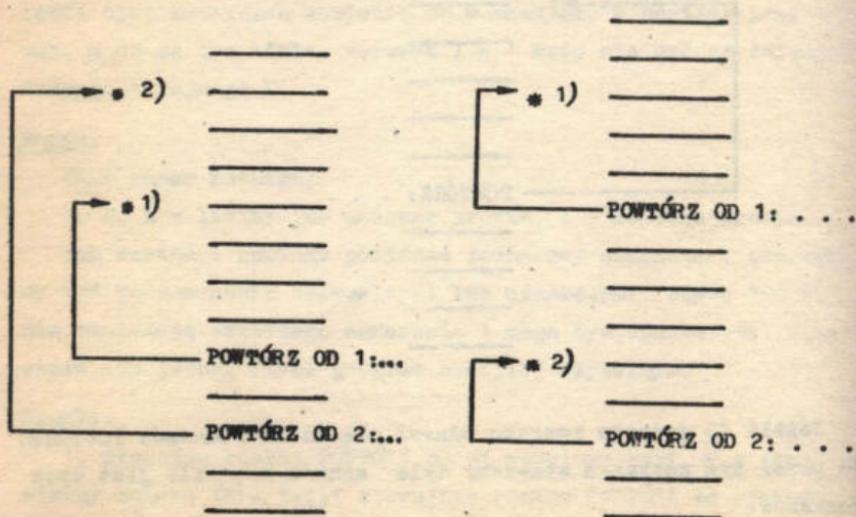
POWTÓRZ: . . .

Jeżeli do jednego rozkazu odnosi się kilka rozkazów POWTÓRZ, to przed tym rozkazem stawiamy tyle znaków *, ile jest tych rozkazów:

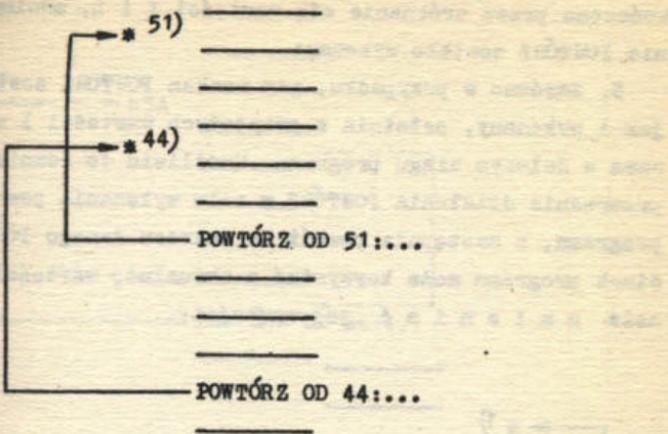


2. Rozkaz POWTÓRZ może się odnosić tylko do rozkazu, wypisanejgo przed nim w programie.

3. Dwa zasięgi mogą być albo całkowicie rozbaczne /tzn. nie posiadać żadnego wspólnego rozkazu/, albo jeden z nich może zawierać się wewnątrz drugiego:

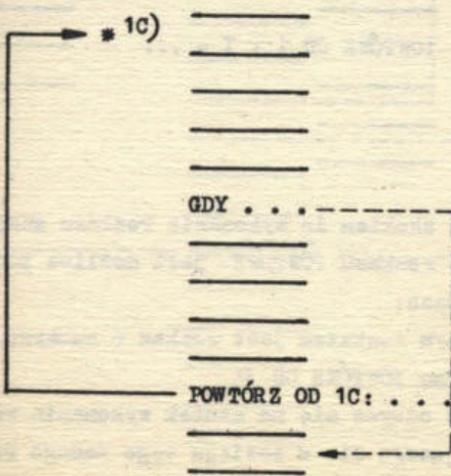


Nie jest prawidłowy więc następujący układ rozkazów POWTÓRZ:



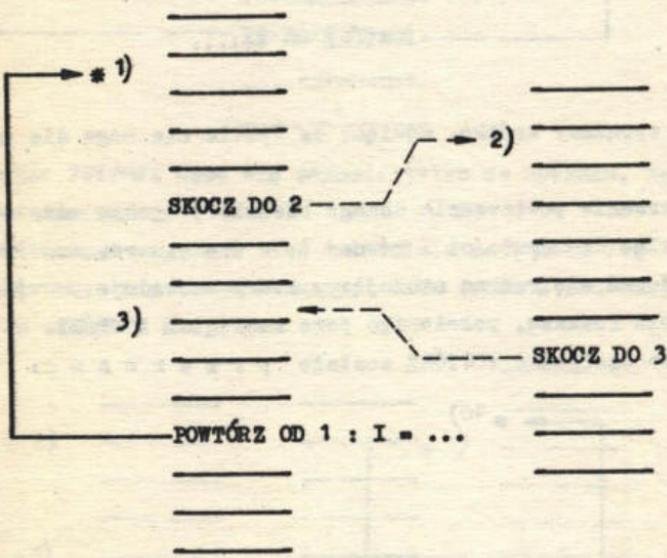
Zasadę tę wyrażamy krótko, mówiąc, że "pętle nie mogą się przecinać".

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



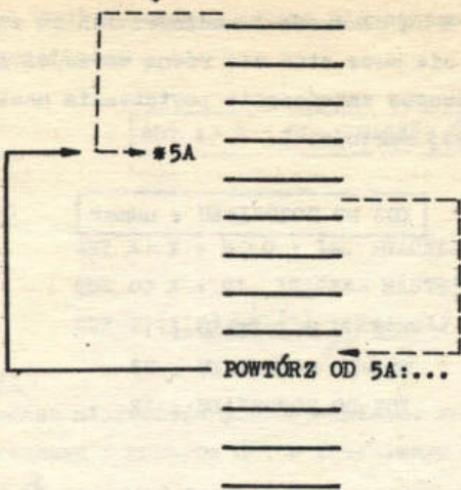
Jeśli natomiast powtarzanie zasięgu rozkazu POWTÓRZ zostało zakończone przez zrównanie się wartości I i L, mówimy, że działanie POWTÓRZ zostało wykonane.

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

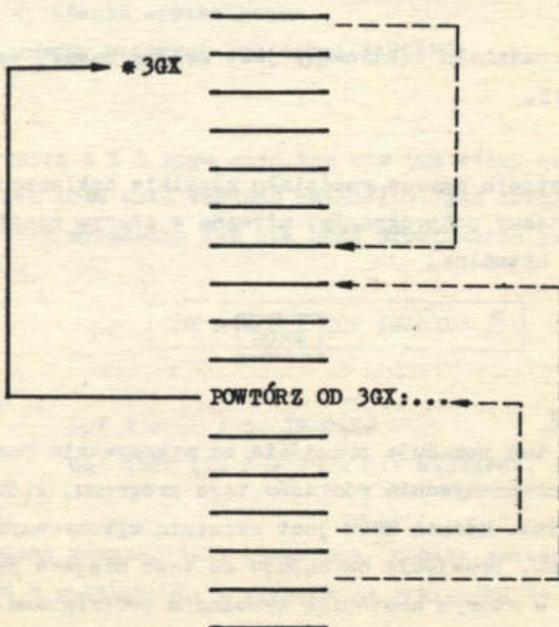


6. Przejście skokiem do wykonania rozkazu znajdującego się w zasięgu pewnego rozkazu POWTÓRZ jest możliwe ponadto w następujących przypadkach:

- gdy wykonywanym rozkazem jest rozkaz o numerze A i rozkaz POWTÓRZ ma formę POWTÓRZ OD A
- gdy przejście odbywa się na skutek wykonania rozkazu sterującego, znajdującego się w zasięgu tego samego rozkazu POWTÓRZ;



Poza opisanymi wyżej przypadkami, żadne przejście do wykonania rozkazu z zasięgu POWTÓRZ 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 : numer

Przykłady:

IDZ DO ROZDZIAŁU : 1

IDZ DO ROZDZIAŁU : 23

IDZ DO ROZDZIAŁU : 12

Znaczenie:

Rozkaz powoduje wczytanie z pamięci bębnowej do pamięci wewnętrznej maszyny nowego rozdziału o podanym numerze, a następnie przejście do wykonania pierwszego rozkazu z nowo wczytanego rozdziału.

Forma:

Numer rozdziału zbudowany jest według reguł, podanych na stronie 112.

Uwaga:

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

WRÓĆ

Znaczenie:

Rozkaz ten powoduje przejście od wykonywania rozkazów podprogramu do wykonywania rozkazów tego programu, który ten podprogram wywołał. Rozkaz WRÓĆ jest ostatnim wykonywanym rozkazem podprogramu. Przejście następuje do tego miejsca programu wywołującego, w którym nastąpiło wywołanie podprogramu. Mówiąc krót-

ko, rozkaz WRÓĆ realizuje powrót z programu podległego do programu nadzorowanego.

GDY $A > B : \alpha$, INACZEJ β

Przykłady:

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

GDY $O > X : 81$, INACZEJ NASTĘPNY

GDY $B(I,J) > C(I)$: INACZEJ NASTĘPNY 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 NASTĘPNY, oznacza ono rozkaz wypisany w najbliższej kolejności.

Forma:

A, B - wyrażenia arytmetyczne

α, β - numery rozkazów lub słowa NASTĘPNY.

Uwaga:

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

2. Wartość wyrażenia A-B nie może przekraczać przyjętej skali obliczeń.

GDY $A = B : \alpha$, INACZEJ β

Przykłady:

GDY $A = B1 : 5$, INACZEJ 2

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

Znaczenie:

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

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

Forma:

A, B - wyrażenie 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 słwo NASTEPNY, oznacza one 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 : 2 CD, INACZEJ 3

GDY KLUCZ I : NASTĘPNY, INACZEJ 5X

Znaczenie:

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

Jeśli zamiast α lub β występuje słowo NASTĘPNY, 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 NASTĘPNY.

Klucze wymienione w tym rozkazie, znajdują się na pulpicie operatora maszyny ZAM-2. Są one ponumerowane liczbami od 0 do 35. Każdy z nich ma dwa stany: podniesiony i opuszczony

[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 NASTĘPNY,

Uwaga:

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

3. Rozkazy arytmetyczne i bulowskie

A	=	B
---	---	---

/formuła arytmetyczna/

Przykłady:

$$A = B + C$$

$$X1 = ALFA + LOG (SIN (X))$$

$$I = I + 1$$

$$S = S + A (I, J) \times B (J, K)$$

$$D = 45.6789$$

$$A (I, J + 4) = S + 12.3$$

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

2. Jeżeli 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 prostej A, stojącej po lewej stronie znaku $=$. Formuła bulowska nadaje lub zmienia dotychczasowe wartości zmiennych.

Forma:

A - zmienna ułamkowa lub całkowita /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 dotychczasowej 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.

3. Formuła bulowska posiada również znaczenie deklaracji, gdyż nadaje odpowiednie znaczenie działaniom, stojącym po prawej stronie formuły.

4. W formule bulowskiej nie mogą występować zmienne indeksowane.

(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 wyrażeniach arytmetycznego, podczas gdy formuła operacyjna nadaje wartości układowi zmiennych poprzez wykonanie działań podprogramu na układzie jego argumentów. W miejsce 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 podstawionym argumentem. Tak więc zamiast pisać

$$(A, B) = \text{TRAP}(C, D, ., ., ., ., .)$$

można krócej pisać:

$$(A, B) = \text{TRAP}(C, D)$$

2. Podstawione argumenty zachowują swoją wartość aż do wykonania następującego 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 (., ., SS)

PODSTAW: CAŁKA (., 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 podprogramu.

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, podstawionych do podprogramu, uzupełnia się

parę 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 DZIESIĘTNIE: I

Przykłady:

USTAW SKALE DZIESIĘTNIE: 3

USTAW SKALE DZIESIĘTNIE: 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 wykonania rozkazów zawierających zmienne ułamkowe, musi być poprzedzone rozkazem USTAW SKALE DZIESIĘTNIE lub USTAW SKALE BINARNIE. Ustalona skala zachowuje swoje znaczenie przy przejściu do podprogramów oraz przy przejściu z jednego rozdziału do drugiego.

USTAW SKALE BINARNIE: I

Przykłady:

USTAW SKALE BINARNIE: 5

USTAW SKALE BINARNIE: ALFA

Znaczenie:

Rozkaz ten ustala skalę dwójkową wszystkich zmiennych ułamkowych 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 DZIESIĘTNIE.

ZWIĘKSZ SKALE DZIESIĘTNIE O I: lista

Przykłady:

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

ZWIĘKSZ SKALE DZIESIĘTNIE O - 7 : EF, * G

Znaczenie:

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

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 DZIESIĘTNIE/.

Uwagi:

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

Przykład:

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

Rozkaz: ZWIĘKSZ SKALE DZIESIĘTNIE 0 - 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 odpowiednio zaokrąglone.

Binarnie
ZWIĘKSZ SKALE DZIESIĘTNIE 0 I : lista

Przykłady:

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

ZWIĘKSZ 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 ..., +34, +35

Uwaga:

Analogicznie, jak w przypadku ZWIĘKSZ SKALE DZIESIĘTNIE.

4. Rozkazy wejścia i wyjścia

CZYTAJ : lista

Przykłady:

CZYTAJ : AB, * ALFA, OMEGA

CZYTAJ : * AXYS

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. Liczby stanowiące dane wejściowe mogą być zaopatrzone w znak + lub -.

Forma:

Lista - zbudowana jest ze zmiennych prostych i nazw bloków.

Reguły:

1. Każda liczba, przewidziana jako wartość zmiennej prostej, jak i każdy układ liczb, stanowiący blok, musi zaczynać się od nowego wiersza formularza danych wejściowych.
2. Po ostatniej liczbie zapisanej w bloku następować musi oddzielnny 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 znakami⁷ jednej liczby nie może występować spacja. Kolejność wczytywania liczb, wypisanych w jednym wierszu, jest od lewej do prawej strony formularza.
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

⁷ Znaki liczby: + lub -, cyfry, kropka pozycyjna.

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 DZIESIĘT-NIE lub USTAW SKALE BINARNIE. Jeżeli pewna wielkość wczytywana nie mieści się w przewidzianej skali, maszyna przerywa obliczenia wykazując błąd.

Uwaga:

1. 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 nazwa bloku o 25 elementach, rozkaz

CZYT AJ: A, B, = C

spowoduje wczytanie

$$1 + 1 + 25 = 27$$

liczb do pamięci maszyny.

2. Nienapisanie gwiazdki * przed nazwą bloku powoduje potraktowanie danej zmiennej jako symbolu pojedynczej liczby.

CZYT AJ WIERSZ : lista

Przykłady:

CZYT AJ WIERSZ : ALFA, BETA

CZYT AJ WIERSZ : N

Znaczenie:

Rozkaz ten powoduje wczytanie do pamięci wewnętrznej maszyny kolejnych znaków, począwszy od skrajnie lewego znaku najbliższego wiersza, tworzącego dane wejściowe. Wartość każdego znaku /patrz tablica 1 na s. 14/ jest wpisywana jako liczba całkowita do każdego z kolejnych słów bloku o podanej nazwie. Znaki są wczytywane do znaku P.K. /Powtó Karetki/ włącznie.

Forma:

Lista składa się z nazw bloków, zadeklarowanych uprzednio jako całkowite.

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 /rozdz. 1, § 4, pkt b/.

Uwaga:

Ostatnim znakiem wczytywanego wiersza jest zawsze Powrót Kartki /P.K./

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 rozumieemy 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)

Znaczenie:

Rozkaz powoduje kolejne wypisanie w aktualnie drukowanym 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 arkusza 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 -. Początko-

wo zarezerwowane pozycje, nie zajęte cyframi znaczącymi liczby i znakiem liczby, wypełnione 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 znajduje się w I+1 zarezerowanej 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 dane zapisane rozkazami DRUKUJ, TEKST i SPACJA nie przekroczyły pojemności jednego wiersza, która wynosi 69 znaków.

DRUKUJ WIERSZ : lista

Przykłady:

DRUKUJ WIERSZ : ALFA, BETA

DRUKUJ WIERSZ : M

Znaczenie:

Rozkaz ten powoduje wypisanie wiersza począwszy od skrajnie lewej pozycji arkusza w ten sposób, że każdy kolejny znak daleko-

pisowy jest określony przez sześć najmniej znaczących pozycji kolejnego słowa bloku o podanej nazwie. Zakończenie drukowania następuje do znaku P.K. /Powrót Karetki/ wyłącznie.

Forma:

Lista składa się z nazw bloków, zadeklarowanych uprzednio jako całkowite.

Reguła:

Ilość słów w bloku, a tym samym ilość wypisanych znaków dalekopisowych nie może przekraczać 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 słowa 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 kropki stawianych co 3 znaki. Łęcznie, na wydrukowanie słowa zarezerwowanych jest 15 pozycji.

W przypadku gdy zmienna 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 RÓWNAŃ LINIOWYCH

METODA ITERACJI

TEKST WIERSZY 1:

TABLICA WARTOŚCI FUNCKJI 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 I

Przykłady:

SPACJA 3

SPACJA

SPACJA BETA

Znaczenie:

Rozkaz powoduje pozostawienie I spacji w aktualnie wypisywanym wierszu.

Forma:

I - liczba naturalna lub zmienna prosta, całkowita, o wartości naturalnej.

Uwaga:

Zamiast pisać SPACJA 1 można krócej pisać SPACJA.

LINIA I

Przykłady:

LINIA 5

LINIA 8

LINIA N

LINIA

Znaczenie:

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

Forma:

I - jak w rozkazie SPACJA

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 /69/ ilości znaków mieszczących się w jednym wierszu formularza.

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

3. Rozkaz LINIA I można krócej pisać: LINIA.

Uwaga:

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

5. Rozkazy współpracy z bębnem

CZYTAJ Z BĘBNA OD I : lista

Przykłady:

CZYTAJ Z BĘBNA OD 120: A, * BETA

CZYTAJ Z BĘBNA OD 2134: X, A, -

R, B, * DZETA, KSI

CZYTAJ Z BĘBNA 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:

.. 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:

CZYT AJ Z BĘBNA OD 120: *A, *B, C

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

120
121
•
•
•
144

} blok A

145
146
•
•
•
204

} blok B

205 zmienna C

2. Kolejność wypisania zmiennych na liście rozkazu CZYTAJ Z BEBNA musi być zgodna z kolejnością zapisu liczb na bębnie.
3. Nienapisanie gwiazdki * przed nazwą bloku powoduje potraktowanie danej nazwy jako zmiennej prostej.

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 bębnowej. Liczby te wpisuje się do pamięci bębnowej 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 dodatniej,

lista - jest utworzona ze zmiennych prostych i nazw bloków.

Uwaga:

Analogiczna, jak w rozkazie CZYTAJ Z BEBNA.

K)

6. Komentarze

Przykłady:

- K) PIERWIASTEK JEST UJEMNY
- K) PRZYPADEK, GDY X-GAMMA JEST > 0
- K) BADANIE, CZY WARUNEK ZBIEŻNOŚCI JEST SPEŁNIONY

Znaczenie:

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

Forma:

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

Rozdział 5

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 zawczasu 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.

1. Terminologia i oznaczenia

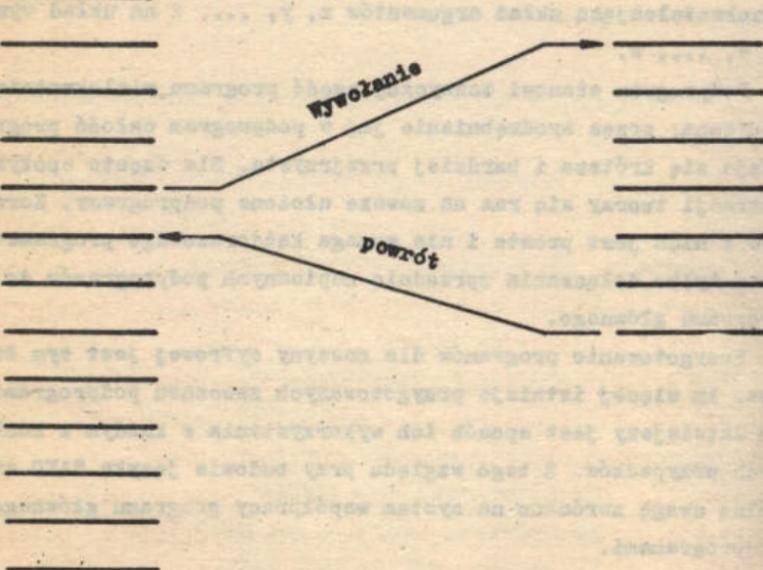
W punkcie tym opisane są pewne pojęcia i terminy, używane w dalszym ciągu.

Przejście do wykonania podprogramu nazywamy wywołaniem podprogramu. W trakcie obliczeń jeden podprogram bywa zazwyczaj wywoływany kilkakrotnie.

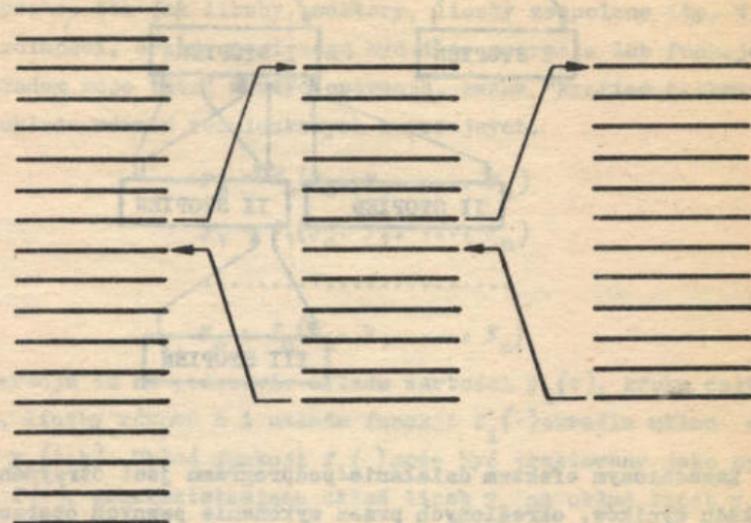
Program, wywołujący podprogram, nazywa się nadziednym w stosunku do danego podprogramu; podprogram wywoływany nosi nazwę podległego w stosunku do programu nadziednego.

Przejście do wykonywania programu nadziednego nazywa się powrotem.

nadrzędny podległy



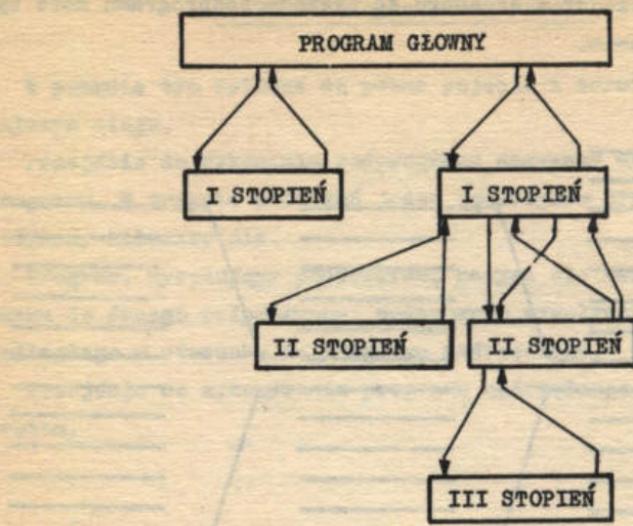
Program, nadzędny w stosunku do pewnego podprogramu może być sam podprogramem.



Zachodzi wówczas przypadek tzw. wielostopniowego korzystania z podprogramów.

Część programu, która nie jest podległa żadnej innej części, nosi nazwę programu głównego. Każdy rozdział programu pisaneego 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 podstawiania.

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 $y_1(t)$, kroku całkowania h , liczby równań n i układu funkcji $f_i()$ określa układ wartości $y_1(t+h)$. Układ funkcji $f_i()$ może być traktowany jako operacja $f()$, przekształcająca układ liczb y_1 na układ liczb y_1 . 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 odpowiednio w punkcie 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 pierwszego 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 funkcją całkową. Przykładem funkcjonowania 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:

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

Operacja ta jest więc typu:

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

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.

2. Ogólne właściwości 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 wynikiem jest jedna liczba,
- możliwość używania jako argumentów podprogramu
liczb,
zmiennych,
bloków liczbowych,
operacji,
- możliwość otrzymywania w wyniku
liczb,
bloków liczbowych.

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 WRÓĆ, który realizuje powrót do programu nadzędnego. Tak więc jedyną komunikację między programem nadzędnym a podległym są z jednej strony rozkazy PODSTAW oraz formuły arytmetyczne i operacyjne, które powodują podstawienie argumentów i wywołują program podległy /była o nich mowa na s. 136, 138, 139/ z drugiej zaś strony rozkaz WRÓĆ.

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 programu; podprogram taki, posiadający niezależną symbolikę, można dołączać do każdego programu głównego. Również wszelkie deklaracje programu nadzędnego /z wyjątkiem deklaracji dotyczącej skali/ tracą moc w stosunku do programu podległego.

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

PODPROGRAM: (U, V) = TRANS (X, Y, ALFA)
S C = COS (ALFA)
C = SIN (ALFA)
U = X × C + Y × S
V = -X × S + Y × C
WRÓĆ

Jeśli operacja jest funkcją, to znaczy posiada tylko jeden wynik liczbowy, wówczas forma budowy podprogramu, definiującego tę funkcję 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 podprogramu jest rozkaz

- nazwa funkcji () = odp. wyrażenie arytmetyczne

lub:

nazwa funkcji () = odp. wyrażenie bulowskie.

Przykład:

PODPROGRAM: WIELOMIAN (X, * A, N)

CAŁKOWITE: N, I

STRUKTURA (N): A

S = 0

* 1) S = S * Y + A(I)

POWTORZ OD 1 : I = (N - 1) 0

WIELOMIAN() = S

WRÓĆ

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

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

CAŁKOWITE: 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

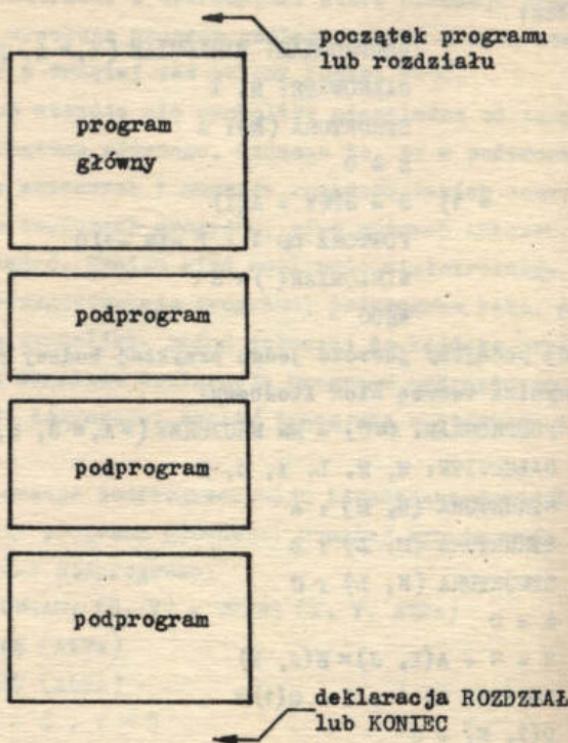
POWTORZ OD 2 : I = 0(1)N

POWTORZ OD 2 : K = 0(1)L

WRÓĆ

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 podstawić 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.

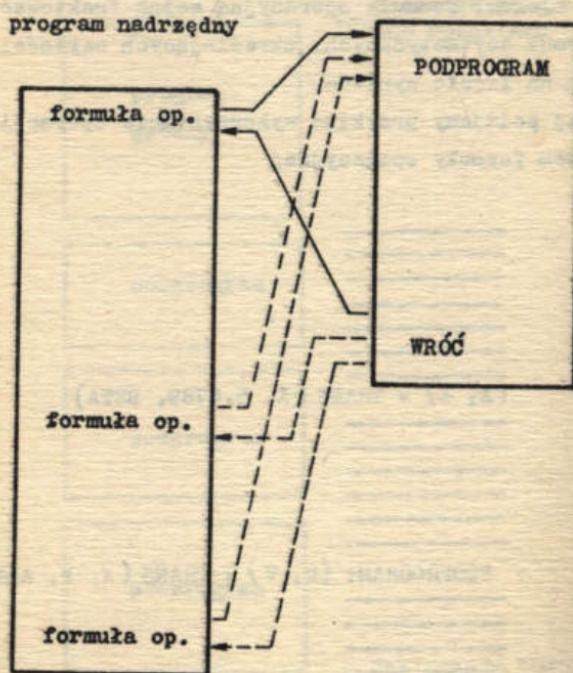
Poniżej podajemy przykład wykorzystania operacji TRANS za pośrednictwem formuły operacyjnej

(X, Y) = TRANS (X, 5.6789, BETA)

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

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

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



Jeśli podprogram definiuje funkcję, wówczas korzystamy z niego za pośrednictwem formuły arytmetycznej według takich samych zasad, jak w 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)

WRÓĆ

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

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

Postać rozkazu PODSTAW jest następująca:

PODSTAW: f (lista podstawionych argumentów)

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

PODSTAW: TRANS (. , H, .)

(B, C) = TRANS (3.456, . , D)

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

WRÓĆ

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:

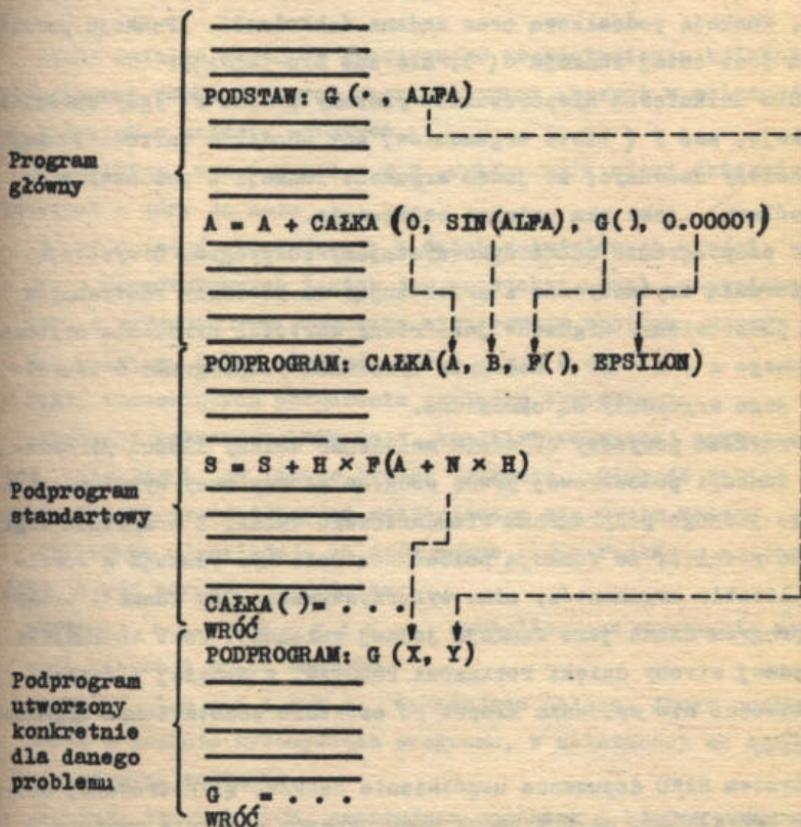
PODSTAW: TRANS (A)

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

$$(U, V) = \text{TRANS} (.)$$

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 CAŁKA jest funkcją, G() jest funkcją. Podprogram CAŁKA jest podprogramem II rzędu, podpro-

gram (G) jest podprogramem I rzędu. Argumentami podprogramu CAŁKA 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 funkcji G jako równy ALFA. Następnie poprzez formułę arytmetyczną wywołuje podprogram CAŁKA, 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 zauważyc, że jeden argument funkcji G pozostał nieokreślony - jest nim zmienna całkowania.

W programie CAŁKA 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 CAŁKA. Z przykładu tego widać również, że funkcją podcałkową może być funkcja o dowolnej liczbie argumentów, mimo wykorzystywania tej funkcji przez podprogram CAŁKA jako funkcji jednej zmiennej. Jest to możliwe z jednej strony dzięki rozkazowi PODSTAW, z drugiej - dzięki możliwości nie wpisania kropki po ostatnim podstawionym argumentem.

System SAKO dopuszcza uogólnienie zasad, zilustrowanej powyższym przykładem, na przypadek podprogramów dowolnie wysokiego rzędu, o dowolnej ilości argumentów, które są również podprogramami dowolnie wysokiego rzędu o dowolnej ilości parametrów.

ROZDZIAŁ 6

SIECI DZIAŁAŃ

Sieci działań służą do graficznego przedstawienia kolejności wykonywania poszczególnych operacji przez maszynę w zależności od podanych w zadaniu warunków.

Kierunkiem normalnym wykonywania sieci działań jest kierunek z góry do dołu.

Sieci działań składają się z dwóch zasadniczych części:

- figur geometrycznych takich jak prostokąt, owal, sześciokąt itp. z umieszczonym wewnętrz nich pewnym napisem. Figury takie będziemy dalej nazywać skrzynkami,
- linii stanowiących połączenia pomiędzy skrzynkami.

Skrzynki wskazujące operacje, kreślimy zazwyczaj pewnej stałej odległości od lewego brzegu arkusza. Długość takiej skrzynki zależy od długości znajdującego się w nim napisu.

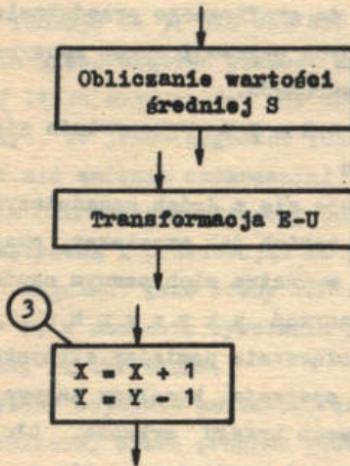
Skrzynkom odpowiadają na ogół pewne grupy rozkazów w języku SAKO. Wśród skrzynek wyróżniamy:

- skrzynki operacyjne, symbolizujące wykonanie pewnych operacji przez maszynę,
- skrzynki logiczne, wskazujące dalszą drogę postępowania w trakcie wykonywania programu, w zależności od spełnienia pewnych warunków.
- skrzynki START i STOP, wskazujące początek i koniec wykonywania programu przez maszynę.

- skrzynki Łącznikowe, służące do identyfikacji przejść pomiędzy skrzynkami lub sieciami działań w przypadku gdy nie jest ono zaznaczone nieprzerwaną linią ciągłą. Połączenia między skrzynkami wskazują na kolejność przejścia z jednej skrzynki do drugiej. W przypadku skrzynek logicznych, posiadających wiele wyjść, właściwe wyjście zależne jest jeszcze od warunków, wskazanych przez taką skrzynkę.

Skrzynki OPERACYJNE ogólnie

Przykłady:



Znaczenie:

Skrzynki operacyjne symbolizują operacje, odpowiadające powyżej grupie zdań w języku SAKQ.

Forma:

Prostokąt z wpisaną nazwą operacji.

Uwagi:

1. W przypadku skrzynek operacyjnych, odpowiadających grupie

zdań SAKO, na ogół nie staramy się wpisać tych zdań wewnętrz skrzynki, a wpisujemy jedynie nazwę tej grupy. Nazwę tę powtarzamy najczęściej w programie w postaci komentarza.

2. W przypadku operacji, np. wejścia i wyjścia posiadających dla przedstawienia skrzynki o szczególnej formie, unikamy zasadystosowania skrzynki operacyjnej ogólnej.

Skrzynki wejścia i wyjścia

Przykłady:



Znaczenie:

Skrzynki te symbolizują operacje wprowadzania lub wyprowadzania danych z maszyny.

Forma:

Skrzynki symbolizujące operacje wejściowe mają formę prostokąta z dodatkową wewnętrzną linią pionową obok lewego boku.

Skrzynki symbolizujące operacje wyjściowe mają formę prostokąta z dodatkową wewnętrzną linią pionową obok prawego boku.

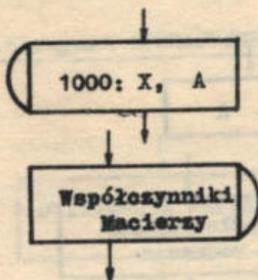
Uwagi:

1. Skrzynka wejścia z wpisaną wewnątrz listą zmiennych symbolizują rozkaz CZYTAJ z tą samą listą.

2. Skrzynki wyjścia z wpisaną wewnątrz listą zmiennych symbolizują odpowiednią grupę zdań takich jak DRUKUJ, LINIA, SPACJA, potrzebnych dla wydania zmiennych wskazanych na liście. Skrzynka wyjściowa może sygnalizować również drukowanie tekstu. Do skrzynki można wówczas wpisać sam tekst lub jego nazwę.

Skrzynki czytania lub pisania na bęben

Przykłady:



Znaczenie:

Skrzynki te symbolizują przesyłanie danych z wewnętrznej pamięci maszyny do bębnowej lub odwrotnie.

Forma:

Skrzynki symbolizujące przesyłanie danych z pamięci wewnętrznej do pamięci bębnowej mają formę prostokąta z dodatkowym kątem zewnętrznym obok prawego boku.

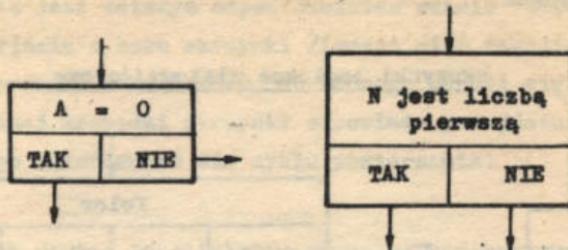
Skrzynki symbolizujące przesyłanie danych z pamięci bębnowej do pamięci wewnętrznej mają formę prostokąta z dodatkowym kątem zewnętrznym obok lewego boku.

Uwagi:

Skrzynki powyższe z wypisaną liczbą całkowitą, dwukropkiem i listą zmiennych symbolizują odpowiednio rozkaz CZYTAJ z BEBNA OD lub PISZ NA BEBNIE OD z tymi samymi danymi.

Skrzynka logiczna alternatywy

Przykłady



Znaczenie:

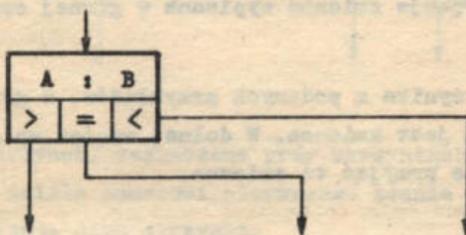
Skrzynka ta wyznacza jedną z dwóch dróg dalszego wykonywania programu w zależności od tego, czy warunek zapisany w górnej części skrzynki jest spełniony czy nie.

Forma:

Wynika z podanych przykładów.

Skrzynka logiczna porównania

Przykład:



Znaczenie:

Skrzynka ta wyznacza jedną z trzech dróg dalszego wykonywania programu w zależności od wyniku porównania dwóch liczb wpisanych w górną część skrzynki.

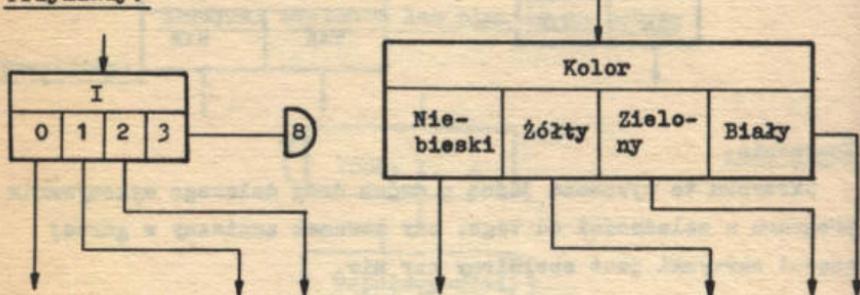
Forma:

Forma graficzna wynika z podanych przykładów. Porównywane wielkości wpisane w górnej części skrzynki przedzielone są dwu-

kropkiem, symbolizującym jeden ze znaków zapisanych w dolnej części skrzynki.

Skrzynki logiczne wielowyjściowe

Przykłady:



Znaczenie:

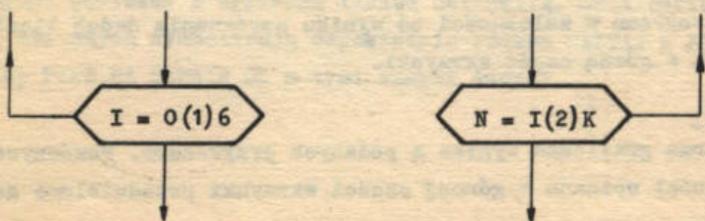
Skrzynka ta wyznacza jedną z wielu dróg dalszego wykonywania programu w zależności od tego, którą z wartości wypisanych w dolnej części przyjmuje zmienna wypisana w górnej części.

Forma:

Forma graficzna wynika z podanych przykładów. W górnej części skrzynki wpisana jest zmienna. W dolnej części wpisane są wartości, jakie może przyjmąć ta zmienna.

Skrzynka powtórz

Przykłady:



Znaczenie:

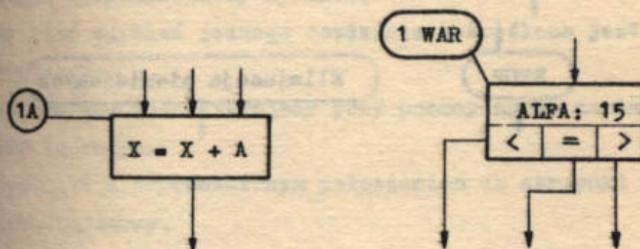
Skrzynka ta jest ścisłym odpowiednikiem zdania POWTÓRZ w języku SAKO. Wyjście z boku skrzynki /lewego albo rzadziej prawego/ powinno prowadzić do numeru, do którego odnosi się to zdanie. Wyjście z dolnej krawędzi skrzynki odpowiada przejściu do następnego zdania po zakończeniu się cyklu powtarzania.

Forma:

Skrzynka ma postać sześciokąta, wewnętrz w której wpisana jest część zdania POWTÓRZ, występująca w nim po dwukropku.

Numeracja skrzynek

Przykłady:



Znaczenie:

Numery skrzynek, założone przy skrzynkach na sieci działań odpowiadają ściśle numerowi pierwszego zdania z grupy zdań SAKO, jaki symbolizuje dana skrzynka.

Forma:

Numer wpisany jest w kółko lub owal pozostający na zewnątrz skrzynki i połączony linią ze skrzynką.

Reguły:

1. Każda skrzynka o więcej niż jednym wyjściu musi być oznaczona numerem.
2. Jeżeli którakolwiek ze skrzynek posiada więcej niż jedno wyjście, wówczas wszystkie za wyjątkiem jednej skrzynki, do których prowadzą te wyjścia, muszą być oznaczone numerami.

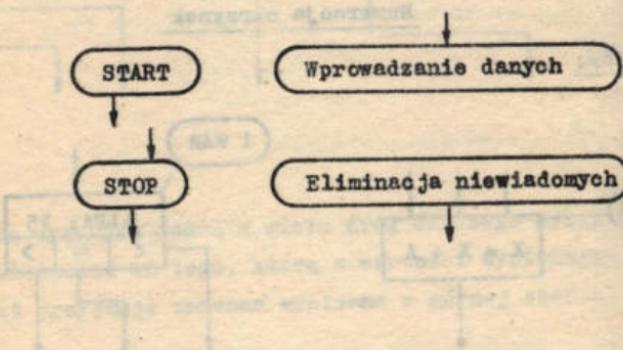
3. Ponadto każda skrzynka może być opatrzona numerem.
Zbędnych numerów należy unikać.

4. Jako numery wolno jedynie stosować symbole przewidziane dla numerowania zdań.

Skrzynki start, stop, rozdział

idź do rozdziału

Przykłady:



Znaczenie:

. Skrzynka z napisem START wskazuje początek sieci działań. Inne skrzynki owalne z wprowadzonym połączeniem oznaczają zdanie ROZDZIAŁ o numerze lub nazwie wpisanej w skrzynkę.

Skrzynka z napisem STOP oznacza zatrzymanie się maszyny. Jest ona ścisłym odpowiednikiem zdania STOP.

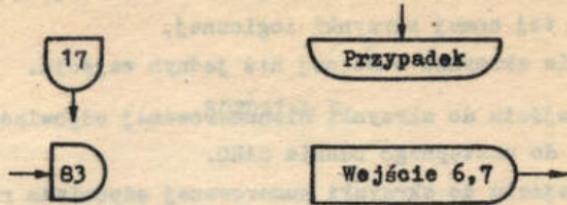
Inne skrzynki owalne z doprowadzonym połączeniem oznaczają zdanie IDŹ DO ROZDZIAŁU o numerze lub nazwie wpisanej w skrzynkę.

Forma:

Wszystkie powyższe skrzynki mają kształt owalny.

Łączniki dla identyfikacji połączeń

Przykłady:



Znaczenie:

Łączniki służą do identyfikacji połączeń w przypadkach, gdy nie są one nakreślone jedną linią ciągłą. Przypadki takie mogą w szczególności zachodzić:

- na jednym arkuszu, gdy nakreślenie pełnego połączenia linią ciągłą komplikowałoby rysunek,
- gdy sieć działań jednego rozdziału nakreślona jest na kilku arkuszach.

Połączenia identyfikujemy przy pomocy nazw, napisanych wewnątrz łącznika.

Łączniki z doprowadzonym połączeniem do skrzynki oznaczają początek przerwy.

Łączniki z wyprowadzonym połączeniem ze skrzynki oznaczają koniec przerwy.

Forma:

Skrzynka ta ma postać prostokąta z zaokrąglonymi dolnymi lub bocznymi rogami i z wpisaną wewnątrz i wolną nazwą.

Uwagi o przekładzie sieci działań na język SAKO

Przekład sieci działań na program w języku SAKO może być na ogół dokonany na wiele sposobów. Jednak dla wprowadzenia pewnej jednolitości postępowania, a często i przejrzystości przekładu, zalecane jest przyjęcie następujących reguł postępowania:

1. W sieci działań numerujemy skrzynki tylko w przypadkach koniecznych, a więc numerujemy
- wszystkie, za wyjątkiem jednej skrzynki, znajdującej się na wyjściu tej samej skrzynki logicznej,
 - wszystkie skrzynki o więcej niż jednym wejściu.
2. Przejściu do skrzynki nienumerowanej odpowiada zawsze przepis do następnego zdania SAKO.
3. Przejściu do skrzynki numerowanej odpowiada rozkaz SKOCZ lub, w przypadku skrzynki logicznej, wypisanie tego numeru w odpowiadającym tej skrzynce zdaniu.
- Po zakończeniu przekładu celowym jest sprawdzenie, czy nie została opuszczona którakolwiek ze skrzynek.

ROZDZIAŁ 7

STRUKTURA SYNTAKTYCZNA JĘZYKA SAKO

Przez strukturę syntaktyczną języka rozumiemy ścisły opis formy wszelkich wyrażeń języka. Takiemu opisowi poświęcony jest niniejszy rozdział.

1. Symbolika opisu

Każde wyrażenie SAKO stanowi pewien ciąg znaków dalekopisowych /patrz tablica na str. /, zbudowany według pewnych reguł. Reguły te wyrażamy przy użyciu pewnej prostej symboliki, której objaśnienie jest celem tego paragrafu.

Niech A, B będą pewnymi, dowolnymi wyrażeniami /moga to być pojedyncze znaki dalekopisowe/.

Wprowadzamy następujące oznaczenia:

1. Oznaczać będziemy przez

$$\left\{ \begin{array}{l} A \\ B \end{array} \right\}$$

wyrażenie, które powstaje przez wybór jednego z wyrażeń A, B

2. Oznaczać będziemy przez

AB

wyrażenie, które powstaje przez napisanie wyrażenia B bezpośrednio po wyrażeniu A.

Oznaczenia te w naturalny sposób uogólniamy na dowolną ilość wyrażeń, tak więc na przykład

A B C D

oznacza wyrażenie powstałe przez kolejne bezpośrednie wypisanie po sobie wyrażeń A, B, C i D.

Dowolną kombinację oznaczeń 1 i 2 nazywać będziemy strukturą.

Przykład 1.

Struktura następująca:

SKALA $\left\{ \begin{array}{l} \text{DZIESIETNA} \\ \text{BINARNA} \end{array} \right\}$ PARAMETROW

oznacza dowolne z wyrażeń:

SKALA DZIESIETNA PARAMETROW

lub

SKALA BINARNA PARAMETROW

Przykład 2.

Struktura:

A $\left\{ \begin{array}{c} B \\ C \\ B \end{array} : \left\{ \begin{array}{c} D \\ DE \end{array} \right\} \quad F \right\}$

oznacza dowolne z następujących wyrażeń:

AB

AC

AB : DF

AB : DEF

W opisie używać będziemy poza samymi wyrażeniami także naz-

wy wyrażeń. Dla odróżnienia wyrażenia od nazwy wyrażenia, te ostatnie zamykać będziemy w nawiasy <, >.

Przykład 3.

Struktura:

$$\text{ABC} \left\{ \begin{array}{l} \text{D } < \text{e} \rangle \\ < \text{f} \rangle \end{array} \right\}$$

oznacza wyrażenie, powstałe przez dopisanie do wyrażenia ABC wyrażenia D i wyrażenia o nazwie e, lub przez dopisanie do wyrażenia ABC wyrażenia o nazwie f.

3. Przez zapis symboliczny

$$< \text{a} \rangle :: = \text{B}$$

rozumieć będziemy, że wyrażenie B ma nazwę a. Zapis taki nazywać będziemy definicją.

Przykład 4.

Definicja

$$< \text{g} \rangle :: = \text{ABC} \left\{ \begin{array}{l} \text{D } < \text{e} \rangle \\ < \text{f} \rangle \end{array} \right\}$$

mówiąc, że wyrażenie o nazwie g powstaje przez dopisanie do wyrażenia ABC albo wyrażenia D i wyrażenia o nazwie e albo wyrażenia o nazwie f.

Przykład 5.

$$< \text{a} \rangle :: = \left\{ \begin{array}{l} \text{B} \\ < \text{a} \rangle \text{ B} \end{array} \right\}$$

Definicja ta mówi, że wyrażenie B ma nazwę a oraz że wyrażenie, powstałe przez dopisanie do wyrażenia o nazwie a wyrażenia B ma również nazwę a. Które więc z następujących wyrażeń:

B
BB
BBB
.....
BBB... B
.....

ma nazwę a.

4. Zamiast pisać:

$$\left\{ \begin{array}{l} A \\ A B \end{array} \right\}$$

piszemy krócej:

$$A [B]$$

i podobnie zamiast pisać:

$$\left\{ \begin{array}{l} A \\ B A \end{array} \right\}$$

piszemy:

$$[B] A$$

Symboli [,] wskazują na możliwość opuszczenia zawartego w nich wyrażenia.

Przykład 6.

Struktura:

$$A \quad \left\{ \begin{array}{l} B \quad [: \quad D \quad [E] \quad F] \\ C \end{array} \right\}$$

określa te same wyrażenia, co struktura podana w przykładzie 2, to znaczy:

$$A B$$

$$A B : D F$$

$$A B : D E F$$

$$A C$$

Przykład 7.

Definicja:

$$\langle a \rangle ::= [\langle a \rangle] B$$

mówi, tak samo jak w przykładzie 5, że a jest nazwą każdego z wyrażeń:

B
 B B
 B B B

 B B B ... B

5. Zamiast pisać:

$$/1/ \quad \langle a \rangle ::= \left\{ \begin{array}{c} B \\ \langle a \rangle , B \end{array} \right\}$$

piszemy:

$$/2/ \quad \langle a \rangle ::= [B]$$

Ponieważ definicja 1 mówi, że nazwę a ma bądź wyrażenie B bądź wyrażenie, powstałe z a przez dopisanie pary znaków:, B zatem struktura:

$$[B]$$

oznacza wyrażenie postaci:

$$B, B, B, \dots, B$$

Zwróćmy uwagę, że definicję /1/ można, na mocy punktu 4, zapisać w postaci:

$$\langle a \rangle ::= [\langle a \rangle ,] B$$

Zauważmy też, że po obu stronach znaku :: = występuje nazwa definiowana.

Przykład 8.

Struktura:

BLOK([<całkowita>])

oznacza dowolne wyrażenie postaci:

BLOK(<całkowita>, <całkowita>, ..., <całkowita>)

6. Dla skrócenia opisu, zamiast wyrażenia, zbudowanego z dwóch znaków dalekopisowych: "P.K." /powrót karetki, w tablicy 1 na str. wartość 30/ i "LINIA" /nowa linia, w tablicy 1 wartość 13/ piszemy symbol

(KL)

który odczytujemy: karetka, linia. Jeśli w strukturze występuje więc znak (KL), to oznacza on przejście do początku kolejnego wiersza formularza.

Przy użyciu oznaczeń, wprowadzonych w punktach 1, 2, 3, 4, 5 i 6 opisujemy strukturę syntaktyczną języka SAKO, to jest formalny opis wszelkich jego wyrażeń.

2. Opis wyrażeń SAKO

Zgodnie z paragrafem poprzednim, opis języka SAKO zawiera szereg definicji, określających jego wyrażenia. Definicje te są pomumerowane i przy każdej nazwie, występującej w strukturze, podany jest numer definicji, określającej wyrażenie o tej nazwie, to jest definiującej daną nazwę. Ułatwia to odszukanie żądanej definicji.

Pewne wyrażenia nie są zdefiniowane według zasad paragrafu 1, lecz sposób ich budowy nie nastręcza wątpliwości; są to:

spacja

czyli odstęp między znakami /znak nieistotny w programie SAKO, lecz istotny jako separator danych wejściowych/, oraz

< ciąg znaków, nie zawierający znaku: ani =>

i

< ciąg znaków, nie zawierający KL >

których budowa wynika z nazwy.

Ponadto wyrażenie o nazwie < program w SAS, 66 > jest zdefiniowane przez odwołanie się do kompendium programowania w języku SAS. Opis tego wyrażenia wykracza poza ramy podręcznika.

Poniżej podane są definicje wyrażeń języka SAKO, poczynając od definicji programu, jako najogólniejszej, a kończąc na definicjach cyfry i litery, jako najbardziej szczegółowych. Zdefiniowane również formę danych wejściowych /def. 40/, które chociaż nie wchodzą w skład programu SAKO, występują niemal w każdym konkretnym problemie obliczeniowym.

1. <program, 1> ::= $\left[\begin{array}{l} \langle \text{ciag rozdzialow}, 2 \rangle \\ \langle \text{rozdzial}, 3 \rangle \end{array} \right]$ KONIEC $\left[: \langle \text{caalkowita}, 56 \rangle \right]$ $\textcircled{K1}$
2. <ciag rozdzialow, 2> ::= $\left[\begin{array}{l} \langle \text{ciag rozdzialow}, 2 \rangle \\ \text{KONIEC ROZDZIAŁU} \end{array} \right]$ ROZDZIAŁ: <caalkowita, 56> $\textcircled{K2}$ <rozdzial, 3>
3. <rozdzial, 3> ::= <ciag zdani, 4> <ciag podprogramow, 37>
4. <ciag zdani, 4> ::= <ciag zdani, 4> <zdanie, 5>
5. <zdanie, 5> ::= <numer, 55> $\left[\begin{array}{l} \text{deklaracja, 6} \\ \text{rozkaz, 7} \\ \left\{ \begin{array}{l} \text{zdanie powtorz, 23} \\ \text{komentarz, 26} \end{array} \right\} \end{array} \right]$ $\textcircled{K3}$

$\left\{ \begin{array}{l} \text{BLOK}(\langle \text{całkowita}, 56 \rangle) \\ \text{STRUKTURA}(\langle \text{nazwa}, 54 \rangle) \end{array} \right\}) : \left[\begin{array}{l} - \text{K1} \\ \langle \text{nazwa}, 54 \rangle \end{array} \right]$
 $\left. \begin{array}{l} \text{TABLICA}(\langle \text{całkowita}, 56 \rangle) : \langle \text{nazwa}, 54 \rangle \text{ K1} \langle \text{blok liczbowy}, 43 \rangle \text{ K1} * \\ \text{TABLICA OCTALNA}(\langle \text{całkowita}, 56 \rangle) : \langle \text{nazwa}, 54 \rangle \text{ K1} \langle \text{blok oktalny}, 48 \rangle \text{ K1} * \end{array} \right\}$
 $6.<\text{deklaracja}, 6> ::=$
 $\left\{ \begin{array}{l} \text{CAŁKOWITE} : \left[\begin{array}{l} - \text{K1} \\ \langle \text{nazwa}, 54 \rangle \end{array} \right] \left[\begin{array}{l} \langle \text{nazwa}, 54 \rangle \\ () \end{array} \right] \\ \text{SKALA} \left\{ \begin{array}{l} \text{DZIESIĘTNIA} \\ \text{BINARNA} \end{array} \right\} \text{ PARAMETROW} : \langle \text{całkowita}, 56 \rangle \\ \text{JĘZYK SAS} \text{ K1} \langle \text{program w SAS}, 65 \rangle \end{array} \right\}$

$\left\{ \begin{array}{l} \langle \text{rozkaz sterujący}, 8 \rangle \\ \langle \text{rozkaz arytmetyczny}, 9 \rangle \\ \langle \text{formula bulowska}, 10 \rangle \\ \langle \text{rozkaz wejścia 1 wyjścia}, 11 \rangle \\ \langle \text{rozkaz współpracy z bębnem}, 12 \rangle \end{array} \right\}$
 $7.<\text{rozkaz}, 7> ::=$

1. $\left\{ \begin{array}{l} \text{SKOCZ} \left\{ \begin{array}{l} \text{WEDŁUG } \langle \text{nazwa}, 54 \rangle : \llbracket [- \odot] \left\{ \begin{array}{l} \langle \text{numer}, 55 \rangle \\ \text{NASTĘPNY} \end{array} \right\} \rrbracket \\ \text{STOP} \left\{ \begin{array}{l} \langle \text{numer}, 55 \rangle \\ \text{NASTĘPNY} \end{array} \right\} \end{array} \right\} \right\}$
2. $\langle \text{rozkaz sterujący}, \text{\&8} \rangle ::= \left\{ \begin{array}{l} \text{IDZ DO ROZDZIAŁU:} \left\{ \begin{array}{l} \langle \text{nazwa}, 54 \rangle \\ \langle \text{osalkowita}, 56 \rangle \end{array} \right\} \\ \langle \text{wyr. arytm. 27} \rangle \left\{ \begin{array}{l} \langle \text{numer}, 55 \rangle \\ \text{INACZEJ} \left\{ \begin{array}{l} \langle \text{numer}, 55 \rangle \\ \text{NASTĘPNY} \end{array} \right\} \end{array} \right\} \end{array} \right\}$
3. $\langle \text{rozkaz sterujący}, \text{\&8} \rangle ::= \left\{ \begin{array}{l} \text{GDY BYŁ NADMIAR} \\ \quad \langle \text{nazwa}, 54 \rangle \\ \quad \text{KLIUCZ} \left\{ \begin{array}{l} \langle \text{nazwa}, 54 \rangle \\ \langle \text{osalkowita}, 56 \rangle \end{array} \right\} \\ \quad \langle \text{formula arytmetyczna}, 13 \rangle \end{array} \right\}$
4. $\langle \text{rozkaz arytmetyczny}, \text{\&9} \rangle ::= \left\{ \begin{array}{l} \langle \text{formula operacyjna}, 14 \rangle \\ \langle \text{rozkaż skalujący}, 17 \rangle \end{array} \right\}$
5. $\langle \text{formula arytmetyczna}, 13 \rangle$
6. $\langle \text{formula arytmetyczna}, 13 \rangle$

11. <roskaz wejścia i wyjścia, 11> ::= $\left\{ \begin{array}{l} \langle \text{roskaz ozyt.j., 18} \rangle \\ \langle \text{roskaz drukuj, 19} \rangle \\ \langle \text{roskaz wydawniczy, 21} \rangle \end{array} \right\}$

12. <roskaz współpracy z bieżnem, 12> ::= $\left\{ \begin{array}{l} \text{CZYTAJ Z BEZNA} \\ \text{OD} \left\{ \begin{array}{l} \langle \text{nazwa, 54} \rangle \\ \langle \text{całkowita, 56} \rangle \end{array} \right\} : \llbracket [-\text{K1}] \langle \text{nazwa, 54} \rangle \rrbracket \end{array} \right\}$
 $\left\{ \begin{array}{l} \text{PISZ NA BEZEN} \end{array} \right\}$

13. <formula arytmetyczna, 13> ::= <nazwa, 54> $\llbracket [\llbracket \text{wyraż. arytm., 27} \rrbracket] \rrbracket$ = <wyraż. arytm., 27>

14. <formula operacyjna, 14> ::= $\left\{ \begin{array}{l} \text{PODSTAW:} \\ (\llbracket * \rrbracket < nazwa, 54>) - \end{array} \right\}$
 $\left\{ \begin{array}{l} \langle \text{wyraż. funkcyjne, 15} \rangle \\ - \end{array} \right\}$

15. <wyrażenie funkcyjne, 15> ::= <nazwa, 54> $(\llbracket \left\{ \begin{array}{l} \langle \text{nazwa, 54} \rangle \\ \langle \text{pusty argument, 16} \rangle \end{array} \right\} \rrbracket)$

16. <pusty argument, 16> ::= ..

- $$17. \langle \text{rozkaz skaluj} \rangle ::= \begin{cases} \text{USTAW SKALE} \left\{ \begin{array}{l} \text{DZIESIETNE} \\ \text{BINARNE} \end{array} \right\} : \left\{ \begin{array}{l} \langle \text{nazwa}, 54 \rangle \\ \langle \text{całkowite}, 56 \rangle \end{array} \right\} \\ \text{ZWIEKSZ SKALE} \left\{ \begin{array}{l} \text{DZIESIETNE} \\ \text{BINARNE} \end{array} \right\} : \left[\begin{array}{l} \langle \text{nazwa}, 54 \rangle \\ [-] \langle \text{całkowita}, 56 \rangle \end{array} \right] \end{cases} : \left[\begin{array}{l} [- \textcircled{K}] \\ \langle \text{nazwa}, 54 \rangle \end{array} \right]$$
- $$18. \langle \text{rozkaz czytelj. 18} \rangle ::= \text{CZYTAJ} \left\{ \begin{array}{l} \text{OKTALNIE} \\ \text{WIERSSZ} \end{array} \right\} : \left[\begin{array}{l} [- \textcircled{K}] \\ [*] \end{array} \right] \left[\begin{array}{l} \langle \text{nazwa}, 54 \rangle \\ \langle \text{nazwa}, 54 \rangle \end{array} \right]$$
- $$19. \langle \text{rozkaz drukuj, 19} \rangle ::= \text{DRUKUJ} \left\{ \begin{array}{l} \text{OKTALNIE} \\ \text{WIERSSZ} \end{array} \right\} : \left[\begin{array}{l} \langle \text{nazwa}, 54 \rangle \\ \langle \text{całkowita}, 56 \rangle \end{array} \right] \left[\begin{array}{l} \langle \text{nazwa}, 54 \rangle \\ \langle \text{całkowita}, 56 \rangle \end{array} \right] \right\} : \langle \text{lista drukuj, 20} \rangle$$
- $$20. \langle \text{lista drukuj, 20} \rangle ::= \langle \text{nazwa}, 54 \rangle \left[\begin{array}{l} + \langle \text{całkowita}, 56 \rangle \\ \langle \text{całkowita}, 56 \rangle \end{array} \right] \right\}$$
- $$21. \langle \text{rozkaz wydawniczy, 2} \rangle ::= \begin{cases} \text{TEKST} : \textcircled{K} \langle \text{tekst}, 51 \rangle \\ \text{TEKST WERSZY} \langle \text{całkowita}, 56 \rangle : \textcircled{K} \langle \text{ciag wierszy}, 53 \rangle \\ \text{SPACJA} \left\{ \begin{array}{l} \text{LINTA} \end{array} \right\} [: \left\{ \begin{array}{l} \langle \text{nazwa}, 54 \rangle \\ \langle \text{całkowite}, 56 \rangle \end{array} \right\}] \end{cases}$$

22. <zasieg powtórz, 21> ::= $\left\{ \begin{array}{l} \langle zdanie powtórz, 23 \rangle \\ \langle numer, 55 \rangle \end{array} \right\}$ <zasieg zdan, 4>

23. <zdanie powtórz, 23> ::= * <zasieg powtórz, 22> <rozkaz powtórz, 24>

24. <rozkaz powtórz, 24> ::= POWTÓRZ [0D <numer, 55>] : <nazwa, 54> = <lista powtórz, 24>
25. <lista powtórz, 25> ::= $\left\{ \begin{array}{l} \langle całkowita, 56 \rangle \\ \langle nazwa, 54 \rangle \end{array} \right\}$ { $\left[\begin{array}{l} \langle całkowita, 56 \rangle \\ \langle nazwa, 54 \rangle \end{array} \right]$ }) $\left\{ \begin{array}{l} \langle całkowita, 56 \rangle \\ \langle nazwa, 54 \rangle \end{array} \right\}$
 $\left\{ \begin{array}{l} \langle użamkowa, 57 \rangle \\ \langle nazwa, 54 \rangle \end{array} \right\}$ { $\left[\begin{array}{l} \langle użamkowa, 57 \rangle \\ \langle nazwa, 54 \rangle \end{array} \right]$ }) $\left\{ \begin{array}{l} \langle użamkowa, 57 \rangle \\ \langle nazwa, 54 \rangle \end{array} \right\}$

26. <komentarz, 26> ::= K) <tekst, 51>

27. <wyraż.arytm., 27> ::= [<wyraż. arytm., 27> { + } { - }] <iloraz, 28>

28. <iloraz, 28> ::= [<iloraz, 28> /] <iloczyn, 29>

29. <iloczyn, 29> ::= [<iloczyn, 29>] <potęga, 30>

30. <potęga, 30> ::= [<potęga, 30> *] <proste wyraż. arytm., 31>

30. <proste wyraz. arytm. 30> ::=

{
 <czalkowite, 56>
 <udzialkowa, 57>
 <zmienna, 32>
 {<wyrazenie funkcyjne, 15>
 <wyrazenie arytmetyczne, 27>
 }
}

31. <proste wyraz. arytm. 31> ::=

{
 <zmienna, 32>
 {<wyrazenie funkcyjne, 15>
 <wyrazenie arytmetyczne, 27>
 }
}

32. <zmienna, 32> ::= <nazwa, 54> [[<wyraz. arytm., 27>]]]

33. <wyraz. bulowski, 33> ::= [<wyraz. bulowski, 33> +] <koniunkcja, 34>

34. <koniunkcja, 34> ::= [<koniunkcja, 34> \times] <przesuniecie, 35>

35. <przesuniecie, 35> ::=

{
 <proste wyraz. bulowski, 36>
 {<nazwa, 54>
 {<czalkowite, 56>
 }
}

36. <proste wyraz. bulowski, 36> ::=

{
 <slowo bulowskie krotkie, 60>
 {<slowo bulowskie długie, 61>
 {<nazwa, 54>
 {<proste wyraz. bulowski>
 {<wyrat. bulowskie, 33>
 }
 }
}

37. <ciag podprogramow, 37> ::= [<ciag podprogramow, 37>] <podprogram, 38>

38. <podprogram, 38> ::= <nagłówek podprogramu, 39> $\textcircled{K1}$ <ciało programu, 40>
39. <nagłówek podprogramu, 39> ::= PODPROGRAM : [[[*] nazwa, 54 >]] <nazwa, 54 >]]
40. <dane wejściowe, 40> ::= [<dane wejściowe, 40>] < blok wejścia, 42 > $\textcircled{K2}$ n] $\textcircled{K1}$
41. <jedn. wejście, 40> ::= { <komentarz wejścia, 47> } $\left[\begin{array}{l} \{ + \} \langle \text{cząstka}, 56 \rangle \\ \{ - \} \langle \text{uzamkowa}, 57 \rangle \end{array} \right] \left\{ \begin{array}{l} \langle \text{słowo bułowskie krótkie}, 60 \rangle \\ \langle \text{słowo bułowskie długie}, 61 \rangle \end{array} \right\}$
42. <blok wejściowy, 42> ::= { <blok liczbowy, 43 > } $\left\{ \begin{array}{l} \langle \text{blok oktalny}, 48 \rangle \end{array} \right\}$
43. <blok liczbowy, 43> ::= { <blok całkowitych, 44 > } $\left\{ \begin{array}{l} \langle \text{blok ułamkowych}, 45 \rangle \end{array} \right\}$
44. <blok całkowitych, 44> ::= [<blok całkowitych, 44 >] < separator wejścia, 46 > [+] [-] < całkowita, 56 >]
45. <blok ułamkowych, 45> ::= [<blok ułamkowych, 45 >] < separator wejścia, 46 > [+] [-] < całkowita, 57 >]

46. $\langle \text{separat wj\acute{e}d\acute{c}}, 46 \rangle ::= \left\{ \begin{array}{l} \langle \text{spacja} \rangle \\ \text{\textcircled{K1}} \end{array} \right\} \left[\begin{array}{l} \langle \text{komentarz wej\acute{c}c} \\ \text{znaku : ani -} \end{array} \right] \left\{ \begin{array}{l} \vdots \\ = \end{array} \right\}$
47. $\langle \text{komentarz wej\acute{c}c}, 47 \rangle ::= \langle \text{littera}, 64 \rangle \langle \text{ci\acute{e}g znaków, nie zawi\acute{e}rający znaku : ani -} \rangle \left\{ \begin{array}{l} \vdots \\ = \end{array} \right\}$
48. $\langle \text{blok oktalny}, 48 \rangle ::= \left\{ \begin{array}{l} \langle \text{blok krótkich}, 49 \rangle \\ \langle \text{blok długich}, 50 \rangle \end{array} \right\}$
49. $\langle \text{blok krótkich}, 49 \rangle ::= \left[\begin{array}{l} \langle \text{blok krótkich}, 49 \rangle \\ \langle \text{słowo bulowskie krótkie}, 60 \rangle \end{array} \right]$
50. $\langle \text{blok długich}, 50 \rangle ::= \left[\begin{array}{l} \langle \text{blok długich}, 50 \rangle \\ \langle \text{słowo bulowskie długie}, 61 \rangle \end{array} \right]$
51. $\langle \text{tekst}, 51 \rangle ::= \langle \text{ci\acute{e}g znaków, nie zawi\acute{e}rający \text{\textcircled{K1}}} \rangle$
52. $\langle \text{wiersz}, 52 \rangle ::= \langle \text{tekst}, 51 \rangle \text{\textcircled{K1}}$
53. $\langle \text{ci\acute{e}g wierszy}, 53 \rangle ::= \left[\begin{array}{l} \langle \text{ci\acute{e}g wierszy}, 53 \rangle \\ \langle \text{wiersz}, 51 \rangle \end{array} \right]$
54. $\langle \text{nazwa}, 54 \rangle ::= \left\{ \begin{array}{l} \langle \text{littera}, 64 \rangle \\ \langle \text{nazwa}, 54 \rangle \end{array} \right\} \left\{ \begin{array}{l} \langle \text{littera}, 64 \rangle \\ \langle \text{cyfra}, 62 \rangle \end{array} \right\}$
55. $\langle \text{numer}, 55 \rangle ::= \left\{ \begin{array}{l} \langle \text{cyfra}, 62 \rangle \\ \langle \text{numer}, 55 \rangle \end{array} \right\} \left\{ \begin{array}{l} \langle \text{littera}, 64 \rangle \\ \langle \text{cyfra}, 62 \rangle \end{array} \right\}$

56. całkowita, 56 ::= [[[[<cyrfa, 62>] <cyrfa, 62>] <cyrfa, 62>] <cyrfa, 62>] <cyrfa, 62>

57. <ulokowa, 57> ::= { <ciaz cyfr, 58> · [<ciaz cyfr, 58>] }
{ [<ciaz cyfr, 58>] [<ciaz cyfr, 58>] }

58. <ciaz cyfr, 59> ::= [<ciaz cyfr, 58>] <cyrfa, 62>

59. <snak okt., 59> ::= [.] <cyrfa okt., 63>

60. <szlово bulowskie krótkie, 60> ::= <cyrfa okt., 63> <snak okt. 59> <snak okt. 59>
<snak okt. 59> <snak okt. 59>

61. <szlово bulowskie długie, 61> ::= <szlово bulowskie krótkie, 60> [.] <szlово bulowskie krótkie, 60>

62. <cyrfa, 62> ::= { 0
1
2
3
4
5
6
7
8
9 }

e) Definiując należy unieważnić następujące warunki: Suma ilości cyfr obu ciągów nie może przekroczyć dziesięciu.

63. <cyfra okt., 63> :: =

{
0
1
2
3
4
5
6
7}

64. <litera, 64> :: =

{
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z}

65. <program SAS, 65>, patrz Kompendium programowania w języku SAS.

ROZDZIAŁ 8

SYSTEM OPERACYJNY

W niniejszym rozdziale omówione będą czynności związane z wprowadzeniem do maszyny i uruchamianiem programów SAKO.

1. Przygotowanie taśmy z programem

Przygotowanie taśmy z programem odbywa się przy pomocy urządzenia zwanego dalekopisem. Urządzenie to posiada klawiaturę, podobną do klawiatury maszyny do pisania i pozwala na drukowanie tekstów złożonych z tak zwanych znaków dalekopisowych /patrz s. 13/. Jednocześnie z tekstem drukowanym urządzenie to produkuje taśmę dziurkowaną w ten sposób, że uderzenie w dowolny klawisz powoduje wydziurkowanie na taśmie odpowiednio /dla danego klawisza/ kombinacji dziurek na taśmie i przesunięcie taśmy o jeden rządek naprzód.

Nazwijmy programem taśmowym tekst, który wypisujemy na dalekopisie, w celu uzyskania taśmy z programem SAKO. Otóż struktura tego tekstu, posługując się symboliką wprowadzaną w rozdziale 7, winna być zgodna z definicjami podanymi na s. 200.

Struktura oraz znaczenie pozostałych pojęć użytych w powyższych definicjach opisane są w poprzednich rozdziałach.

Jak wynika z definicji programu taśmowego istnieją dwa sposoby przygotowywania taśmy z programem SAKO.

$\langle \text{program taśmowy} \rangle ::= \left\{ \begin{array}{l} \langle \text{program} \rangle \\ \langle \text{program operacyjny} \rangle \end{array} \right\}$

$\langle \text{program} \rangle ::= \left\{ \begin{array}{l} \langle \text{rozdział} \rangle \\ \langle \text{ciąg rozdziałów} \rangle \end{array} \right\}$ KONIEC $\left[:\langle \text{całkowita} \rangle \right] \textcircled{1}$

$\langle \text{program operacyjny} \rangle ::= \left\{ \begin{array}{l} \langle \text{rozdział} \rangle \\ \langle \text{operacyjny ciąg rozdziałów} \rangle \end{array} \right\}$ KONIEC $\left[:\langle \text{całkowita} \rangle \right] \textcircled{1}$

$\langle \text{operacyjny ciąg rozdziałów} \rangle ::= \left\{ \begin{array}{l} \langle \text{operacyjny ciąg rozdziałów} \rangle \text{ KONIEC ROZDZIAŁU} \\ \langle \text{dyrektywa L} \rangle \text{ ROZDZIAŁ :} \langle \text{całkowita} \rangle \textcircled{1} \langle \text{rozdział} \rangle \end{array} \right\}$

$\langle \text{dyrektywa L} \rangle ::= L [1] [2] [\bullet] [4] [P] \textcircled{1}$

1. Według tekstu o strukturze programu
2. Według tekstu o strukturze programu operacyjnego.

Jak zobaczymy później, wybór jednej z tych możliwości wpływa na sposób wprowadzania informacji z taśmy do maszyny.

Łatwo zauważyc, że program operacyjny jest to po prostu ciąg rozdziałów SAKO, z których każdy z wyjątkiem ostatniego zakończony jest napisem KONIEC ROZDZIAŁU, a przed każdym rozdziałem /ściślej przed ewentualnym napisem ROZDZIAŁ, poprzedzającym dany rozdział/ umieszczona jest dyrektywa L.

Uwaga: Wiersz tekstu, w którym występuje dyrektywa L. nie może zawierać "spacji".

2. Znaczenie dyrektywy L

Dyrektyna L jest informacją, że umieszczone za nią na taśmie informacje stanowią rozdział programu napisanego w języku SAKO. Jest ona jednocześnie rozkazem dla maszyny aby rozdział wprowadzić, przetłumaczyć na język maszyny i wykonać pewne dodatkowe czynności wyznaczone przez ewentualne znaki dalekopisowe występujące za literą L. Poszczególne z tych znaków oznaczają żądanie wykonania różnych czynności. A mianowicie:

Znak '1' - wyprowadzenie, za pośrednictwem perforatora, tak zwanego słownika symboli przyporządkowanego wszystkim symbolom /zmiennych, numerów symbolicznych, podprogramów/ użytym w danym rozdziale odpowiadających im adresów w pamięci wewnętrznej maszyny. Przykład takiego słownika podany jest na s. 219.

Znak '2' - wyprowadzenie, za pośrednictwem perforatora, wprowadzonego rozdziału w języku maszyny /w postaci rozkazów z adresami bezwzględnymi/.

Znak '*' - wyprowadzenie, poprzez perforator, wprowadzonego rozdziału w systemie binarnym, oraz, jeżeli za znakiem '*' w

dyrektywie L występuje znak '4', również wprowadzenie na taśmie binarnej słownika symboli. /Słownik symboli wypisywany jest na taśmie w postaci fikcyjnego rozdziału binarnego o numerze 131066./

Znak '4' - przechowanie w maszynie, wraz z wprowadzonym rozdziałem, słownika symboli. /Wykonanie tej czynności przy wprowadzaniu programu, pozwoli stosować przy jego uruchamianiu dyrektywę J - patrz 4.2.1./.

Znak 'P' - jeżeli wprowadzony rozdział był ostatnim rozdziałem programu /rozdziałem, za którym występował napis KONIEC/ - zatrzymanie maszyny przed rozpoczęciem wykonywania programu. Jeżeli wprowadzony rozdział nie był końcowym - znak 'P' nie powoduje wykonania żadnej czynności.

3. Wprowadzanie programu SAKO do maszyny

Wprowadzanie programów do maszyny wymaga pewnego obeznania z obsługą urządzeń wejścia i wyjścia oraz stolika operacyjnego.

Na niezbędne minimum umiejętności dotyczących obsługi urządzeń wejścia i wyjścia składają się:

- a. Umiejętność prawidłowego zakończenia taśmy do czytnika
- b. Umiejętność wypuszczenia z perforatora odcinka pustej taśmy.

W stosunku do stolika operacyjnego wymagane jest:

- odnalezienie na płycie czołowej stolika, trzech przycisków zaopatrzonych w etykietki: OTP, START, STOP oraz umiejętność ich naciskania

- odnalezienie na płycie czołowej rzędu 36 kluczy tzw. k 1 u-
c z y K₁ oraz umiejętność ustawienia każdego z nich w dwóch po-
łożeniach: "podniesiony" i "opuszczony".

Klucze K₁ ponumerowane są kolejno od prawej strony numerami od 0 do 35.

Wszystkie te umiejętności są natury czysto praktycznej i naj-
prościej je zdobyć przy bezpośrednim kontakcie z maszyną.

Zakładając opanowanie powyższych umiejętności podamy szczegółowy przepis postępowania przy wprowadzaniu programów SAKO do maszyny. Przepis podany będzie w postaci ciągu czynności, które należy wykonać kolejno w celu wprowadzenia napisanego programu do maszyny:

Czynność pierwsza

Przygotowanie taśmy z programem zgodnie z zasadami opisanymi w paragrafie 1.

Czynność druga

Zakończenie przygotowanej taśmy do czytnika

Czynność trzecia

Czynność ta zależy od tego, czy taśma została przygotowana według tekstu o strukturze <program>, czy też <program operacyjny>.

W przypadku programu operacyjnego to znaczy gdy na taśmie występują dyrektywy L/ czynność trzecia polega na ustawieniu wszystkich kluczy K₁ na stoliku w położeniu "podniesiony".

W przypadku gdy taśma została przygotowana według tekstu o strukturze <program>/to znaczy nie zawiera dyrektyw L/, należy ustawić klucz 0 w pozycji "opuszczony", oraz jeżeli chcemy, aby po wprowadzeniu każdego rozdziału programu była wykonywana kolumna z czynnościami opisanymi w paragrafie 2, należy wstawić dodatkowo w położeniu "opuszczony" odpowiedni klucz. Odpowiedniość między znakami oznaczającymi w dyrektywie L poszczególne czynności a numerami kluczy jest przy tym następująca:

- znak '1' - klucz nr 19
- znak '2' - klucz nr 20
- znak '*' - klucz nr 21
- znak '4' - klucz nr 22
- znak 'P' - klucz nr 34

Czynność czwarta

Naciśnięcie przycisku OTP na stoliku operatora.

Efektem czynności czwartej jest uruchomienie maszyny i rozpoczęcie przez nią czytania taśmy założonej do czytnika. Po przeczytaniu i przetłumaczeniu na język maszyny każdego rozdziału SAKO, maszyna wykonuje czynności wyznaczone w dyrektywie L pochodzącej dany rozdział lub też wyznaczone przez ustawienie kluczy na stoliku.

W czasie wprowadzania i tłumaczenia rozdziału programu napisanego w SAKO maszyna wypisuje, za pośrednictwem perforatora, informacje o błędach /przeważnie syntaktycznych/ wykrywanych przez program tłumaczący. Zdania, w których wykryte zostały tego rodzaju błędy nie są przeważnie tłumaczone na język maszyny, a więc nie ma ich w programie wynikowym. Po wypisaniu informacji o błędzie maszyna zatrzymuje się. Jeżeli chcemy aby wprowadzanie programu było kontynuowane należy nacisnąć przycisk START.

Po wprowadzeniu i przetłumaczeniu ostatniego rozdziału oraz wykonaniu ewentualnych czynności wyznaczonych przez parametry dyrektywy L zostaje wypisana liczba wskazująca ile miejsca na bieżnie pozostaje do dyspozycji programu⁸. /Część bębna, od końca zajęta jest przez sam program/. Następnie maszyna przechodzi do wykonywania wprowadzonego programu lub zatrzymuje się, w zależności od treści ostatniej dyrektywy L lub ustawienia kluczy K₁. W przypadku gdy maszyna zatrzyma się przed przejściem do wykonania programu, naciśnięcie przycisku START spowoduje, że maszyna zacznie wykonywać wprowadzony program.

Jeżeli w czasie wprowadzania programu maszyna sygnalizowała pewne błędy, wówczas na ogół nie ma sensu wykonywać wprowadzonego programu, gdyż z góry wiadomo, że jego działanie będzie nieprawidłowe. Należy więc w takim przypadku błędy poprawić i wprowadzanie programu powtórzyć. Może się jednak zdarzyć, że nie

⁸ Liczba ta jest podwojoną liczbą miejsc na bieżnie pozostających do dyspozycji programu.

wszystkie rozdziały programu zawierały błędy. W takim przypadku, w celu przyśpieszenia wprowadzania programu, wskazane jest zastąpienie na taśmie rozdziałów bez błędów odpowiednimi rozdziałami binarnymi. Tego rodzaju zastępowanie można zastosować tylko w przypadku, gdy taśma z programem w SAKO jest zbudowana na zasadzie programu operacyjnego. Wymiana taka polega na wycięciu z taśmy z programem w SAKO odpowiedniego rozdziału łącznie z dyrektywą L poprzedzającą ten rozdział oraz napisem KONIEC ROZDZIAŁU na końcu i wklejeniu na to miejsce taśmy binarnej wyprodukowanej przez maszynę w ramach wykonywania czynności "*" po wprowadzeniu odpowiedniego rozdziału SAKO. Oczywiście jeżeli odpowiednia dyrektywa L zawierała również znak '4', to taśma binarna będzie również zawierać słownik symboli. Stosując opisane tylko co wymiany, otrzymujemy taśmę z tak zwany programem m i e s z a n y m, którego strukturę dokładnie określają definicje podane na następnej stronie.

Uwaga:

W poniższym wyprowadzeniu przyjmuje się, że poszczególne pojęcia oznaczają odcinki taśmy z wydziurkowanymi odpowiednimi tekstami a nie same teksty.

4. Uruchamianie programu

Gdy program zostanie do maszyny wprowadzony bez sygnalizowania błędów, można pozwolić maszynie na jego wykonanie. Może okazać się, że program nie działa tak jak tego życzył sobie programista. Jest to po prostu program inny niż ten, który zamierzał napisać programista inaczej mówiąc program zawiera, z punktu widzenia programisty, błędy semantyczne. Wykrycie takich błędów może być niekiedy dość trudne. Pewnych informacji ułatwiających wykrycie tych błędów dostarcza niekiedy sama maszyna w trakcie wykonywania programu, np. sygnalizuje, że program żądał wyciąg-

<program niesamany> ::= [<ciąg rozdziałów S>] <rozdział końcowy>
 <ciąg rozdziałów S> ::= [<ciąg rozdziałów S>] <rozdział S>
 <rozdział S> ::= {
 <rozdział SAKO>
 <rozdział binary>
 }
 <rozdział SAKO> ::= <dyrektywa L>ROZDZIAŁ:<calkowita> KL <rozdział> KONIEC ROZDZIAŁU KL
 <rozdział binary> ::= <odcinek tajny wydziurkowanej przez maszynę po wprowadzeniu<rozdziału SAKO>
 saczynającego się od : L * [4] KL .>
 <rozdział końcowy> ::= {
 <rozdział końcowy SAKO>
 <rozdział końcowy binary>
 }
 <rozdział końcowy> :: = [<rozdział końcowy binary>] KONIEC [<calkowite>] KL
 <rozdział końcowy SAKO> ::= <dyrektywa L>ROZDZIAŁ:<calkowita> KL <rozdział> KONIEC [<calkowite>]
 <rozdział końcowy binary> ::= <odcinek taśmy wydziurkowanej przez maszynę po wprowadzeniu<rozdziału
 końcowego SAKO> zaczynającego się od tekstu: L * [4] KL >

nięcia pierwiastka kwadratowego z liczby ujemnej, itp. Na ogólnie jednak informacje te, o ile w ogóle są, są niewystarczające i należy posłużyć się systemem operacyjnym w celu uzyskania większej ilości informacji o programie. Znajduje tutaj zastosowanie tak zwany program awaryjny, którego opis podany będzie niżej. Postępuje się przy tym zwykle według następującego schematu:

Każe się maszynie wykonywać wprowadzony program i po pewnym czasie, gdy dochodzi się do wniosku, że program działa nieprawidłowo, zatrzymuje się maszynę /przez naciśnięcie przycisku STOP/ i następnie kaze się maszynie wypisać na zewnątrz pewne informacje o stanie jej pamięci. Na przykład żąda się wypisania informacji jakie wartości mają w danej chwili zmienne użyte w programie.

Można kazać maszynie wyprowadzić tego typu informacje przez wprowadzenie do niej tak zwanego programu awaryjnego. Taśmę z programem awaryjnym powinno się w zasadzie mieć przygotowaną jeszcze przed przystąpieniem do sprawdzania programu.

Poniżej opisane zostaną struktura, znaczenie oraz sposób wprowadzania i działania programu awaryjnego.

4.1. Struktura programu awaryjnego

Program awaryjny jest to ciąg dyrektyw awaryjnych zakończony dyrektywą Z. Dokładnie jego strukturę opiszemy posługując się symboliką wprowadzoną w rozdziale VII.

$\langle \text{program awaryjny} \rangle ::= [\langle \text{ciąg dyrektyw awaryjnych} \rangle] \langle \text{dyrektywa Z} \rangle$
 $\langle \text{ciąg dyrektyw awaryjnych} \rangle ::= [\langle \text{ciąg dyrektyw awaryjnych} \rangle]$
 $\langle \text{dyrektywa awaryjna} \rangle ::=$

$\left\{ \begin{array}{l} \langle \text{dyrektywa J} \rangle \\ \langle \text{dyrektywa O} \rangle \\ \langle \text{dyrektywa P} \rangle \\ \langle \text{dyrektywa Q} \rangle \\ \langle \text{dyrektywa R} \rangle \\ \langle \text{dyrektywa M} \rangle \\ \langle \text{dyrektywa N} \rangle \\ \langle \text{dyrektywa S} \rangle \\ \langle \text{dyrektywa T} \rangle \end{array} \right\}$

<dyrektywa J> ::= J (KL)
 <dyrektywa O> ::= <adres pw> (KL) <ilość> (KL) <skala> (KL) O (KL)
 <dyrektywa P> ::= <adres beb> (KL) <ilość> (KL) <skala> (KL) P (KL)
 <dyrektywa Q> ::= <adres pw> (KL) <ilość> (KL) 17 (KL) Q (KL)
 <dyrektywa R> ::= <adres beb> (KL) <ilość> (KL) 17 (KL) R (KL)
 <dyrektywa M> ::= <adres pw> (KL) <ilość> (KL) M (KL)
 <dyrektywa N> ::= <adres beb> (KL) <ilość> (KL) N (KL)
 <dyrektywa S> ::= <adres pw> (KL) <ilość> (KL) S (KL)
 <dyrektywa T> ::= <adres beb> (KL) <ilość> (KL) T (KL)
 <dyrektywa Z> ::= Z (KL)
 <adres pw> ::= <całkowita>
 <adres beb> ::= <całkowita>
 <ilość> ::= <całkowita>
 <skala> ::= <całkowita>

Zdefiniowany powyżej syntaktycznie program awaryjny jest tekstem który, w celu wprowadzenia go do maszyny, należy przenieść na taśmę dziurkowaną przy pomocy dalekopisu. Należy przy tym uwzględnić zastrzeżenie, że przy produkcji taśmy z programem awaryjnym nie wolno posługiwać się znakiem "spacja".

4.2. Dyrektywy awaryjne

Dyrektyny awaryjne są to zlecenia podawane maszynie w celu wyprowadzenia z pamięci maszyny różnego rodzaju informacji. Różne dyrektywy powodują wyprowadzenie informacji w różnych postaciach i z różnych rodzajów pamięci.

Poniżej podany będzie opis poszczególnych dyrektyw awaryjnych.

4.2.1. Dyrektywa J

Forma:

J (KL)

Znaczenie:

Powoduje wypisanie poprzez perforator informacji o numerze rozdziału, który był wykonywany w momencie gdy, maszyna została

zatrzymana, oraz wartościach, jakie posiadały w tym momencie wszystkie zmienne danego rozdziału jak również adresy tych zmiennych oraz adresy początków ewentualnych podprogramów występujących w danym rozdziale. W przypadku bloków, wypisana zostaje wartość jedynie zerowego elementu bloku.

Postać w jakiej te wszystkie informacje są wypisywane jest następująca.

a. Pierwszy wiersz zawiera numer rozdziału.

b. Następnie wypisywane są informacje o wartościach parametrów podprogramów. Dla każdego podprogramu, zmienne występujące na liście deklaracji PODPROGRAM jako argumenty lub wyniki są poprzedzone nazwą tego podprogramu z adresem jego początku w pamięci wewnętrznej. Nazwa podprogramu, skrócona do trzech pierwszych liter, jest poprzedzona gwiazdką. Wartości parametrów /tzn. argumentów i wyników/ podprogramów podane są w trzech postaciach:

1. jako para rozkazów,
2. jako liczba całkowita,
3. jako liczba ułamkowa.

Jeżeli parametrem podprogramu jest zmienna pojedyncza korzystamy z postaci 2 lub 3, natomiast w przypadku bloku lub funkcji korzystamy z postaci 1. Mianowicie gdy parametrem jest blok, to adres pierwszego z pary rozkazów wskazuje adres początku bloku, który został do podprogramu podstawiony. Gdy parametrem jest funkcja, to adres podstawionej funkcji odczytujemy również z pierwszego z pary rozkazów,

c. Za informacjami o podprogramach wypisane są informacje o zmiennych rozdziału, przy czym w jednym wierszu wypisane są informacje o jednej zmiennej w kolejności następującej:

nazwa zmiennej skrócona do 4-ch pierwszych liter,
adres tej zmiennej w pamięci wewnętrznej,
wartość zmiennej jako liczba całkowita lub ułamkowa w zależności od rodzaju zmiennej.

Wartości zmiennych ułamkowych przeliczane są na system dziesiętny.

siętny zgodnie ze skalą podaną w programie przez ostatni rozkaz USTAW SKALE.

Uwaga:

Dyrektywę J można stosować pod warunkiem, że przy wprowadzaniu programu SAKO żądano przechowania Słownika Symboli /patrz paragraf 2/.

Przykład:

dla programu:

```
L1*4P
ROZDZIAŁ:6
USTAW SKALE DZIESIETNIE:1
SKALA DZIESIETNA PARAMETROW:1
1)BLOK(4):PIES
CALKOWITE:R
R=6
W=3.8
PIES(0)=6.88
Z=LOS(*PIES,R,W)
STOP 1
PODPROGRAM:LOS(*KOT,K,U)
WROC
KONIEC:6
```

efekt wykonania dyrektywy J jest następujący:

ROZDZIAŁ 6

*LOS	32					
KOT	UA.	950+	SS	0	44982	+0.000020946;
K	SS	6	SS	0	6	+0.0000000028
U	.SN.	614	SW	409+	-26214	+3.7999999998
PIES	950			+6.8799999999		
R	949			6		
W	946			+3.7999999998		
Z	944			+1.5024414063		
KOT	z6			+0.0000209464		
K	z8			+0.0000000028		
U	30			+3.7999999998		

4.2.2. Dyrektywa O

Forma:

<adres pw> KL <ilość> KL <skala> KL 0 KL
parametry <adres pw>, <ilość> i <skala> są liczbami całkowitymi.

Znaczenie:

Powoduje wypisanie z kolejnych długich miejsc pamięci wewnętrznej grupy liczb ułamkowych.

Znaczenie parametrów:

<adres pw> - adres pierwszej liczby wypisywanej /musi być parzysty/.

<ilość> - podwojona ilość liczb

<skala> - skala binarna liczb wypisywanych

Przykład:

Dyrektyna:

```
200
10
14
0
```

powoduje wypisanie 5 liczb ułamkowych z miejsc pamięci wewn.
200-208 w następującej postaci:

WD	200
----	-----

+6270.1309518814
+118.9079799652
+8192.0059556961
+8440.0295791354
+1562.5297546387

4.2.3. Dyrektywa P

Forma:

<adres beb> KL <ilość> KL <skala> KL P KL
parametry <adres beb>, <ilość> i <skala> są liczbami całkowitymi.

Znaczenie:

Powoduje wypisanie z kolejnych miejsc pamięci bębnowej grupy liczb ułamkowych.

Znaczenie parametrów:

<adres beb> - podwojony adres bębnowy pierwszej liczby wypisywanej.

<ilość> - podwojona ilość liczb

<skala> - skala binarna liczb wypisywanych.

Przykład:

Dyrektiva:

2000
12
10
F

powoduje wypisanie 6 liczb ułamkowych z miejsc pamięci bębnowej 1000 - 1005 w następującej postaci:

BD 2000

+0.1233333555
+0.3333333333
+0.1010101011
+0.2222222222
+0.0
+0.0

4.2.4. Dyrektywa Q

Forma:

<adres pw> (KL) <ilość> (KL) 17 (KL) Q (KL)

parametry <adres pw> i <ilość> są liczbami całkowitymi.

Znaczenie:

Powoduje wypisanie z kolejnych krótkich miejsc pamięci wewnętrznej grupy liczb całkowitych.

Znaczenie parametrów:

<adres pw> - adres pierwszej liczby wypisywanej /musi być parzysty/.

<ilość> - ilość liczb wypisywanych /musi być parzysta/

Przykład:

Dyrektywa:

700

6

17

Q

powoduje wypisanie 6 liczb całkowitych z miejsc pamięci wewnętrznej 700 - 705 w następującej postaci:

WK 700

+70349.0
+4174.0
+825.0
+66545.0
+103408.0
+21183.0

4.2.5. Dyrektywa R

Forma:

<adres beb> **(KL)** <ilość> **(KL)** 17 **(KL)** R **(KL)**
parametry <adres beb> i <ilość> są liczbami całkowitymi

Znaczenie:

Powoduje wypisanie z kolejnych miejsc pamięci bębnowej grupy liczb całkowitych. Ponieważ miejsca pamięci bębnowej są długie więc z każdego miejsca pamięci bębnowej wypisane zostają dwie liczby całkowite.

Znaczenie parametrów:

<adres beb> - podwojony adres bębnowy pierwszego miejsca, z którego będą wypisywane liczby

<ilość> - ilość liczb wypisywanych /musi być parzysta/

Przykład:

Dyrektiva:

3500
4
17
R

powoduje wypisanie 4 liczb całkowitych z miejsc pamięci bębornej
1750 - 1751 w następującej postaci

BK 3500

-111.0
+23.0
-17.0
+535.0

4.2.6. Dyrektywa M

Forma:

<adres pw> (KL) <ilość> (KL) M (KL)
parametry <adres pw> i <ilość> są liczbami całkowitymi.

Znaczenie:

Powoduje wypisanie z kolejnych miejsc pamięci wewnętrznej maszyny grupy rozkazów w postaci podanej w opisie maszyny ZAM-2 /patrz: Kompendium programowania w języku SAS - Prace ZAM seria C3/.

Znaczenie parametrów:

<adres pw> - adres pierwszego rozkazu /musi być parzysty/
<ilość> - ilość rozkazów /musi być parzysta/

Przykład:

Dyrektiva:

130
8
M

powoduje wypisanie 8 rozkazów z miejsc pamięci wewnętrznej
130-137 w postaci:

130/
UA 971
SK 979
SK 261 : 132
PP 152
PP 153
NI 0
SK 971 : 136
SK 102

4.2.7. Dyrektywa N

Forma:

<adres beb> (KL) <ilosc> (KL) N (KL)
parametry <adres beb> i <ilosc> sa liczbami całkowitymi.

Znaczenie:

Powoduje wypisanie z kolejnych miejsc pamięci bębnowej grupy rozkazów.

Znaczenie parametrów:

<adres beb> - podwojony adres bębnowy pierwszego miejsca, z którego są wypisywane rozkazy

<ilosc> - ilość rozkazów wypisywanych /musi być parzysta/

Przykład:

Dyrektyna:

1000
6
N

powoduje wypisanie 6 rozkazów z miejsc pamięci bębnowej 500-502 w postaci:

SS	11	:	0
SS	15		
SS	15		
SS	772		
SS	10	:	4
SS	I		

Uwaga: para liczb przedzielonych kropką, wypisana w pierwszym wierszu wyznacza wartość parametru <adres bob>. Wartość tę otrzymujemy przez pomnożenie pierwszej z tych liczb przez 256 i dodanie do wyniku drugiej.

4.2.8. Dyrektywa S

Forma:

<adres pw> (KL) <ilosc> (KL) S (KL)
parametry <adres pw> i <ilosc> są liczbami całkowitymi

Znaczenie:

Powoduje wydziurkowanie zawartości grupy kolejnych słów pamięci wewnętrznej w postaci taśmy binarnej. Otrzymana w ten sposób taśma binarna ma tę własność, że założenie jej do czytnika i naciśnięcie przycisku OTP /przy podniesionych kluczach K₁/ spowoduje operację odwrotną, to znaczy wprowadzenie wyprowadzonych informacji do tych samych miejsc pamięci. /Uwaga: aby maszyna się zatrzymała po wprowadzeniu tych informacji musi być na końcu taśmy binarnej dodziurkowana dyrektywa Z/.

Znaczenie parametrów:

<adres pw> - adres pierwszego słowa /musi być parzysty/,
<ilosc> - ilość słów krótkich /musi być parzysta/.

Przykład:

Dyrektyna:

500
256
S

spowoduje wydziurkowanie na taśmie binarnej zawartości miejsc pamięci wewnętrznej 500-755 włącznie.

4.2.9. Dyrektywa T

Forma:

<adres bęb> \textcircled{KL} <ilosc> \textcircled{KL} T \textcircled{KL}
parametry <adres bęb> i <ilosc> są liczbami całkowitymi

Znaczenie:

Powoduje wydziurkowanie zawartości grupy kolejnych słów pamięci bęбnowej w postaci taśmy binarnej. Otrzymane w ten sposób taśma binarna ma tę właściwość, że założenie jej do czytnika i naciśnięcie przycisku OTP /przy podniesionych kluczach K₁/ spowoduje wprowadzenie wyprowadzonych informacji do tych samych miejsc pamięci, z których były wyprowadzone. /Uwaga: aby maszyna się zatrzymała po wprowadzeniu tych informacji należy dodziurkować na końcu taśmy binarnej dyrektywę Z/.

Znaczenie parametrów:

<adres bęb> - podwojony adres bębnowy pierwszego słowa
<ilosc> - ilość słów krótkich /musi być parzysta/

Przykład:

Dyrektyna:

1000
400
T

spowoduje wydziurkowanie na taśmie binarnej zawartości miejsc na bębień 500-699 włącznie.

4.2.10. Dyrektywa Z

Forma:

Z \textcircled{KL}

Znaczenie:

Powoduje zatrzymanie maszyny po uprzednim zregenerowaniu za-

wartości pamięci wewnętrznej maszyny do stanu takiego jaki był przed wprowadzeniem programu awaryjnego. Również stan dostępnej dla programisty części bębna nie ulega zmianie w czasie działania programu awaryjnego.

4.3. Wprowadzanie i działanie programu awaryjnego

Czynności, które należy wykonać, aby wprowadzić program awaryjny, są identyczne z odpowiednimi czynnościami dla programu operacyjnego /patrz: paragraf 3/.

W czasie wprowadzania i po wprowadzeniu każdej dyrektywy maszyna wykonuje wyznaczone przez nią czynności, po czym wprowadzona zostaje następna dyrektywa i tak dalej, aż do dyrektywy Z, która powoduje zatrzymanie maszyny.

4.4. Przykład programu awaryjnego

Przykładem programu awaryjnego może być następujący program awaryjny skomponowany z przykładów dyrektyw awaryjnych podanych w paragrafie 4.2.:

J
200
10
14
0
2000
12
10
P
700
6
17
Q
3500
4
17
R

I 30
8
11
1000
6
N
500
256
S
1000
400
T
Z

4.5. Przykład Słownika Symboli

Poniżej podany jest Słownik Symboli wypisany przez maszynę po wprowadzeniu programu ze str. 68

ROZDZIAŁ : o

I	0
*WP	60
.WP	70
I	116
*PWK	160
*??D	208
*??C	388
xVLK	153
xVLD	160

U	958
V	956
T	954
ERDC	952
ERDB	950
ERDA	948
Z	946
A	64
B	66
C	68

ŁRDA	944
ŁRDD	942
ŁRDF	940
ŁRDG	938
ŁRDE	936
ŁRDC	934
ŁRDB	932
ŁRDH	930

Objaśnienia:

1. W prawej kolumnie podane są symbole programu, a w lewej adresy przyoróżdżkowanych im miejsc pamięci wewnętrznej.
 2. Pierwszych 9 wierszy za napisem ROZDZIAŁ: o stanowią słownik numerów symbolicznych, funkcji i podprogramów. Nazwy funkcji i podprogramów skrócone do 3-ch pierwszych liter poprzedzone są znakiem *. Symbole * ??D i * ??C oznaczają standartowe podprogramy realizujące odpowiednio rozkazy DRUKUJ i CZYTAJ. Symbole zaczynające się od znaków . i x są specjalnymi numerami symbolicznymi wprowadzonymi przez program tłumaczący. Na początku wypisane są numery symboliczne występujące w programie głównym. Za każdym symbolem podprogramu /aż do następnego symbolu podprogramu/ występują numery symboliczne występujące w tym podprogramie.
 3. Za słownikiem numerów symbolicznych, funkcji i podprogramów występuje słownik zmiennych. Na początku tego słownika podane są zmienne programu głównego. Potem kolejno parametry oraz zmienne podprogramów.
- Zmienne zaczynające się od znaku 'Ł' są zmiennymi roboczymi wprowadzonymi przez program tłumaczący formuły arytmetyczne.



Biblioteka Główna
Wojskowej Akademii Technicznej

II-37149



03-059522-0001