AKADEMIA SZTABU GENERALNEGO
im. gen. broni K. Świerczewskiego

INSTYTUT ORGANIZACJI I TECHNIKI DOWODZENIA
OŚRODEK OBLICZENIOWY

JAWNE

PROGRAMOWANIE W AUTOKODZIE MOST 1

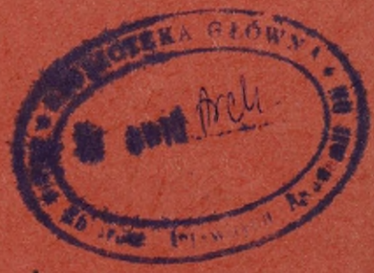
REMBERTÓW GRUDZIEŃ 1965

AKADEMIA SZTABU GENERALNEGO
im. gen. broni K. Świerczewskiego

INSTYTUT ORGANIZACJI I TECHNIKI DOWODZENIA
OŚRODEK OBLICZENIOWY

JAWNE

PROGRAMOWANIE W AUTOKODZIE MOST 1



REMBERTOW GRUDZIEŃ 1965

Autor: mgr Jerzy Szczepkowicz

Wrocławskie Zakłady Elektroniczne ELWRO
Wrocław – ul. Ostrowskiego 30

Korektor: mgr Teodor Mika

NAZWA: **PROGRAMOWANIE W AUTOKODZIE MOST 1**

Data opracowania: 20.XII.1964 r.

03-VI-1

SPIS TREŚCI str.

1. WSTĘP

1.1 Uwagi ogólne

Autokod MOST 1 jest pierwszym systemem programowania automatycznego opracowanym dla maszyny cyfrowej ODRA 1003 w celu istotnego ułatwienia programowania dla tej maszyny. Opracowany autokod jest przeznaczony przede wszystkim do programowania obliczeń naukowych i technicznych, ale w pewnym zakresie nadaje się także do zastosowań administracyjnych.

1.2 Przygotowanie obliczeń i postać wyników

Konstrukcja współczesnych maszyn cyfrowych praktycznie nie pozwala przekazywać im jakichkolwiek informacji pisanych lub drukowanych w zwykłej postaci. Wszystkie informacje, które trzeba przekazać maszynie cyfrowej w celu wykonania obliczeń muszą być zapisane inaczej, zwykle za pomocą znaków dziurkowanych na taśmie papierowej lub kartach papierowych. Maszyna ODRA 1003 wykorzystuje i wydaje informacje wydziurkowane na taśmie papierowej; maszyna dziurkuje taśmy za pomocą szybkiego perforatora, a programy i dane dla maszyny dziurkuje się (i drukuje jednocześnie) za pomocą zwykłych dalekopisów wyposażonych w perforator. Maszyna ODRA 1003 może także drukować wyniki bezpośrednio na podłączonym do niej dalekopisie. Programy, dane i wyniki są drukowane w międzynarodowym kodzie dalekopisowym Nr 2.

1.3 Sposób korzystania z autokodu MOST 1

MOST 1 (zwany dalej krótko autokodem) jest pewną symboliką

zbliżoną nieco do zwykłej symboliki matematycznej i przeznaczoną do opisywania algorytmów obliczeń takich jak rozwiązywanie równania kwadratowego albo całkowanie numeryczne funkcji. Każdy algorytm czyli tzw. program opisany w autokodzie składa się z instrukcji. Są to łatwo zrozumiałe symbole poszczególnych czynności, jakie trzeba wykonać w obliczeniach - działań arytmetycznych, porównywania liczb ze sobą itd.

Można dość łatwo nauczyc się układać programy w autokodzie i czytać je, ale maszyna cyfrowa nie może wykonywać obliczeń bezpośrednio według takich programów. Ma ona bowiem swój język wewnętrzny; jednej instrukcji autokodu odpowiada ciąg co najmniej kilku, a zwykle kilkunastu i więcej instrukcji języka wewnętrznego czyli rozkazów maszyny cyfrowej ODRA 1003. Tłumaczenie programu na język wewnętrzny maszyny, to jest przetworzenie tego programu na ciąg rozkazów realizujących żądane obliczenia, wykonuje maszyna cyfrowa za pomocą programu (ułożonego w języku wewnętrznym), zwanego translatorem. Dla wykonania obliczeń według programu napisanego w autokodzie trzeba wprowadzić do pamięci maszyny translator (wraz z pomocniczymi podprogramami), po czym maszyna cyfrowa może przetłumaczyć dowolny poprawny program (wydziurkowany na taśmie) w języku MOST 1 na swój język wewnętrzny i umieścić ten przekład w swojej pamięci. Natychmiast po zakończeniu tłumaczenia maszyna może wykonać obliczenia opisane tłumaczonym programem, po czym można od razu tłumaczyć następne programy i wykonywać według nich obliczenia. Ta metoda używania translatora nazywa się metodą bezpośrednią (bezpośrednio po przetłumaczeniu programu - jego wykonanie). Zaletą tej metody jest mała liczba

manipulacji z długimi taśmami translatora i pomocniczych podprogramów, tylko raz wprowadzanymi do pamięci maszyny. Metoda bezpośrednia ma jednak tę wadę, że przy jej stosowaniu nie otrzymuje się przekładu utrwalonego na taśmie papierowej, a jedynie jego zapis w pamięci maszyny, który zostanie zniszczony przy tłumaczeniu następnego programu. Ta metoda powoduje znaczną stratę czasu maszyny cyfrowej w tych przypadkach, gdy tłumaczone programy są często używane. Przekład programu wydziurkowany na taśmie papierowej można bowiem wprowadzić do pamięci maszyny kilkadziesiąt razy szybciej niż program autokodowy tłumaczony przez translator. Ponadto, jeżeli program jest bardzo długi lub występuje w nim wiele zmiennych, to nie można go prawidłowo przetłumaczyć tą metodą z powodu braku miejsca w pamięci. W takich przypadkach programy w autokodzie tłumaczy się metodą pośrednią. Stosując ją umieszcza się w pamięci maszyny tylko translator, po czym maszyna może tłumaczyć program w autokodzie i dziurkować przekład na taśmie papierowej. Wydziurkowana w ten sposób taśma nazywa się taśmą dwójkową (albo binarną) programu; treść tej taśmy można szybko umieścić w pamięci maszyny (z szybkością ok. 150 znaków na sekundę).

Po wydziurkowaniu taśmy dwójkowej przez translator można w razie potrzeby tłumaczyć dalsze programy nie wprowadzając ponownie translatora. Przed wykonaniem obliczeń według programu przetłumaczonego w ten sposób, wystarczy wprowadzić do pamięci maszyny taśmę z podprogramami pomocniczymi, a następnie taśmę dwójkową programu. Wprowadzanie taśmy podprogramów pomocniczych można pominąć, jeżeli została ona wprowadzona

do pamięci maszyny wcześniej, np. przy poprzednim korzystaniu z metody bezpośredniej, a programy według których wykonywano obliczenia były poprawne.

Bliższe szczegóły korzystania z translatora można znaleźć w [2]. Prócz powyższych uwag, które mają jedynie zapoznać czytelnika z podstawowymi czynnościami związanymi z posługiwaniem się autokodem MOST 1, niniejszy podręcznik zawiera wyłącznie wiadomości potrzebne przy pisaniu programów w tym autokodzie.

2. WIADOMOŚCI PODSTAWOWE

2.1 Symbole

W programach napisanych w autokodzie można używać dużych liter alfabetu łacińskiego:

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

cyfr:

0 1 2 3 4 5 6 7 8 9

i następujących innych znaków:

-	(minus)
'	(apostrof)
✠	(znak mnożenia)
₥	(znak początku i końca tekstu)
,	(przecinek)
:	(dwukropek)
()	(nawiasy)
+	(plus)
?	(znak zapytania)
.	(kropka)
/	(znak dzielenia)
=	(znak równości)

2.2 Rodzaje liczb

Liczby użyte w programach pisanych w autokodzie, tj. stałe (§2.3) i wartości zmiennych (§2.4) dzieli się na całkowite i zmiennoprzecinkowe. Te ostatnie są na ogół niecałkowite. Liczby mają ograniczoną wielkość: można używać liczb całkowitych z przedziału $\langle -2^{38}, 2^{38}-1 \rangle$, tj. przedziału $\langle -274877906944, 274877906943 \rangle$, oraz liczb zmiennoprzecinkowych z przedziału $\langle -2^{63}, 2^{63}-2^{32} \rangle$, tj. - w przybliżeniu - z przedziału $\langle -9.2\ 10^{18}, 9.2\ 10^{18} \rangle$.

Liczb całkowitych używa się w programach na ogół jako numerów, wskaźników i do podobnych celów organizacyjnych. Działania na tych liczbach maszyna wykonuje dokładnie. Wynik działania na liczbach całkowitych, który nie należy do przedziału dopuszczalnego jest błędny; fakt ten w programach napisanych w autokodzie nie jest sygnalizowany przez maszynę.

Wyniki działań na liczbach zmiennoprzecinkowych maszyna zaokrągla do ok. 9 cyfr dziesiętnych (31 cyfr dwójkowych). Jeżeli wynik działania na liczbach zmiennoprzecinkowych nie należy do przedziału dopuszczalnego, to maszyna zatrzymuje się.

2.3 Stałe

Liczby całkowite występujące w programie pisze się w zwykły sposób. Liczby zmiennoprzecinkowe występujące w programie pisze się w jednej z następujących postaci:

(a) tak, jak liczby całkowite, tzn. bez kropki dziesiętnej; rodzaj liczby bez tej kropki zależy od miejsca liczby w programie; tak napisana liczba zmiennoprzecinkowa nie może przekraczać połowy wartości dopuszczalnej dla liczb całkowitych;

(b) jako co najwyżej 9-cyfrowy ułamek dziesiętny, w którym część całkowitą od części ułamkowej oddziela kropka (przecinka nie można używać do tego celu); zerową część całkowitą można pominąć;

(c) w postaci M'C , gdzie M (mantysa) jest liczbą napisaną w jednej z postaci omówionych powyżej, zaś C (cecha) jest liczbą całkowitą; symbol M'C oznacza liczbę $M \times 10^C$. Jeżeli cecha jest równa zeru, to można ją pominąć.

We wszystkich opisanych postaciach liczb całkowitych i zmiennoprzecinkowych znaku + przed liczbami nieujemnymi nie pisze się.

Przykład 2.3.1 Liczbę $+1.25 \times 10^2$ można napisać w programie w trzech postaciach:

125

125.0 (albo 125.000 itd.)

oraz 1.25'2 (albo 1250'-1 albo .0125'4 itd.)

Dla liczby -1.25×10^2 wszystkie przytoczone postacie mają na początku znak - (minus).

2.4 Zmienne

W programach napisanych w autokodzie można używać zmiennych prostych (tj. bez wskaźników) i zmiennych z jednym wskaźnikiem. Te ostatnie łączy się w tablice o wspólnej nazwie. Zmienne proste i tablice zmiennych z jednym wskaźnikiem oznacza się dużymi literami alfabetu łacińskiego. Nie można używać jednej litery dla oznaczenia i zmiennej prostej, i tablicy. Wszystkie wartości ustalonej zmiennej prostej, a także wszystkie wartości wszystkich zmiennych z ustalonej tablicy muszą być

liczbami jednego typu. Wybór nazw dla użytych zmiennych prostych i tablic (całkowitych i zmiennoprzecinkowych) jest w każdym programie dowolny.

2.5 Umowne znaczenie niektórych liter w podręczniku

W celu skrócenia dalszych definicji przyjmujemy następujące oznaczenia:

I,J,K,L,M,N - zmienne proste i tablice całkowite,

U,X,Y,Z - zmienne proste i tablice zmiennoprzecinkowe.

m,n,p,q - nieujemne stałe całkowite,

i,j,k,l - stałe całkowite,

u,x,y,z - stałe zmiennoprzecinkowe.

Nie obowiązują one poza tym podręcznikiem i programista może użyć np. litery X dla oznaczenia zmiennej całkowitej (§2.4).

2.6 Wskaźniki

Zmienne ze wskaźnikami pisze się inaczej niż w zwykłej symbolice matematycznej. Dopuszczalne postacie zmiennych ze wskaźnikami są następujące:

	Zmienne całkowite		Zmienne zmiennoprzecinkowe	
Zmienne ze wskaźnikami prostymi	Kn		Xn	
	KI		XI	
Zmienne ze wskaźnikami złożonymi	K(I+n)	K(I-n)	X(I+n)	X(I-n)
	K(I+J)	K(I-J)	X(I+J)	X(I-J)
	K(mI)		X(mI)	
	K(mI+n)	K(mI-n)	X(mI+n)	X(mI-n)
	K(mI+J)	K(mI-J)	X(mI+J)	X(mI-J)

Zgodnie z tą tablicą wskaźnik prosty, tj. będący stałą lub zmienną prostą, piszemy z prawej strony nazwy tablicy, bez nawiasów.

Przykład 2.6.1 Symbole A5, I32, JN są zmiennymi ze wskaźnikami prostymi. Należy pamiętać, że symbol JN nie oznacza iloczynu zmiennych J i N, ale zmienną z tablicy J ze wskaźnikiem, który jest wartością zmiennej N.

Wskaźnik złożony, tj. zawierający co najmniej jedno działanie, zamyka się w nawiasy.

Przykład 2.6.2 Symbole A(I-5), X(M-N), K(2J), X(2J-1), X(15K-L) są zmiennymi ze wskaźnikami złożonymi. Symbole K0 lub X0 (zmienna ze wskaźnikiem stałym zero) oznaczają to samo, co odpowiednio litery K lub X (zmienna bez wskaźnika). Symboli takich jak X(2+I), K(5-3I), X(-J+1) nie można używać, ponieważ nie są zgodne z przytoczoną tablicą.

Wskaźniki nie mogą mieć wartości ujemnych. Ponadto wartość zmiennej wchodzącej w skład wskaźnika złożonego, którego drugi składnik jest stałą, nie może być ujemna w chwili, gdy wskaźnik ten jest używany.

We wszystkich dalszych definicjach i przykładach, jeżeli nie powiedziano inaczej, litery U,X,Y,Z,I,J,K lub L oznaczające zmienne można zastąpić dowolną dopuszczalną zmienną (tego samego typu) z dowolnym wskaźnikiem.

2.7 Zewnętrzna postać programów

Każdą instrukcję (§1.3) pisze się w oddzielnym wierszu. Maszyna wykonuje instrukcje w tej kolejności, w jakiej zostały napisane - aż do czasu napotkania jakiejkolwiek instrukcji

zmieniającej tę kolejność.

Przykład 2.7.1 Program, który czyta z taśmy składowe wektora n-wymiarowego, normalizuje je i drukuje po znormalizowaniu, ma następującą postać:

```
INTEGER IN
REAL A1OOST
LABEL 1
BEGIN

1:READ N
S=0
FOR I=1 STEP 1 UNTIL N
READ AI
T=AI*AI
S=S+T
END I
S=SQRT S
PRINTLINE 2
PRINT ŕWEKTOR ZNORMALIZOWANYŕ
FOR I=1 STEP 1 UNTIL N
PRINTLINE 1
T=AI/S
PRINT T,1.8
END I
STOP

START 1
```

Nie będziemy teraz tłumaczyć sensu poszczególnych instrukcji; wspomnimy tylko, że pierwsze cztery instrukcje programu i ostatnia nie są w podanym przykładzie wykonywane przez maszynę w czasie realizacji programu, lecz zawierają jedynie pewne informacje potrzebne do prawidłowego wykonania tłumaczenia przez translator.

3. WYRAŻENIA I INSTRUKCJE ARYTMETYCZNE.

3.1 Funkcje standardowe.

W programach napisanych w autokodzie MOST 1 można używać następujących funkcji standardowych:

(a) funkcji o argumentach i wartościach zmiennoprzecinkowych

ABS - wartość bezwzględna,

SQRT - pierwiastek kwadratowy,

EXP - funkcja wykładnicza z podstawą e,

LN - logarytm naturalny,

SIN - sinus (argument w radianach),

COS - cosinus (argument w radianach),

TAN - tangens (argument w radianach),

ARCSIN - arcus sinus (wartość w radianach),

ARCTAN - arcus tangens (wartość w radianach),

FRAC - część ułamkowa liczby (różnica między argumentem i jego częścią całkowitą czyli ENTIER),

(b) funkcji o argumentach zmiennoprzecinkowych i wartościach całkowitych

ENTIER - część całkowita argumentu,

(przez część całkowitą liczby x rozumie się tutaj największą liczbę całkowitą niewiększą od x i np. ENTIER 4.4 = 4, ENTIER -4.4 = -5)

(c) funkcji o argumentach całkowitych i wartościach zmiennoprzecinkowych

STAND - przekształcenie argumentu do postaci zmiennoprzecinkowej,

(d) funkcji o argumentach i wartościach całkowitych

ABS - wartość bezwzględna.

Argumentem funkcji może być dowolna stała lub zmienna właściwego dla danej funkcji rodzaju. Stałą można poprzedzić znakiem - (minus). Argument funkcji pisze się po prawej stronie nazwy funkcji oddzielony od niej pojedynczym odstępem. Ujmowanie argumentu w nawiasy jest niedopuszczalne.

Przykład 3.1.1 Symbol wartości funkcji wykładniczej w punkcie X152 ma postać EXP X152. Napis EXP -.5 oznacza wartość funkcji wykładniczej w punkcie -.5.

3.2 Wyrażenia arytmetyczne

W autokodzie używa się następujących wyrażeń arytmetycznych

J	X
-J	-X
J⊛K	XØY
-J⊛K	-XØY
FC L	FZ X
-FC L	-FZ X

gdzie

(a) znak ⊛ oznacza jeden z trzech znaków + (dodawanie), - (odejmowanie), ✱ (mnożenie);

(b) znak Ø oznacza jeden z czterech znaków +, -, ✱, / (dzieleni

(c) symbol FC oznacza nazwę dowolnej funkcji standardowej o argumentach całkowitych;

(d) symbol FZ oznacza nazwę dowolnej funkcji standardowej o argumentach zmiennoprzecinkowych.

Każdą ze zmiennych wchodzących w skład wyrażenia arytmetycznego można zastąpić dowolną stałą (§2.3) nieujemną odpowiedniego typu. Znak ✝ na ręcznie pisanych formularzach programu można zastąpić znakiem ×. Pominąć go nie można (jedynie we wskaźnikach -§2.6 - symbol mI ma sens i oznacza iloczyn stałej m przez zmienną prostą I).

Zauważmy, że w autokodzie argumenty działania arytmetycznego muszą być tego samego typu (oba całkowite lub oba zmiennoprzecinkowe). Wielkości całkowitych nie można dzielić. Wynik działania arytmetycznego na wielkościach całkowitych jest dokładny, a wynik działania arytmetycznego na wielkościach zmiennoprzecinkowych jest zwykle przybliżony. Wartość dowolnego wyrażenia z lewej kolumny jest liczbą całkowitą, wartość wyrażenia z prawej kolumny jest liczbą zmiennoprzecinkową. Wyjątkami są wyrażenia zawierające funkcje ENTIER i STAND, dla których rodzaje wartości ustalono w §3.1, (b) i (c).

Przykład 3.2.1 Napisy

```
512
-125'4
-125×K13
ABS KL
3.1415926+X(I+J)
ENTIER X128
-COS Y(2K-5)
-XN+U(J-K)
```

są wyrażeniami arytmetycznymi. Znak - w ostatnim wyrażeniu odnosi się do zmiennej XN; wyrażenie to ma sens taki sam, jak wyrażenie U(J-K)-XN.

3.3 Instrukcje arytmetyczne

Instrukcja arytmetyczna postaci

L=A lub X=B

(gdzie A i B są wyrażeniami arytmetycznymi o wartościach odpowiednio typu całkowitego i zmiennoprzecinkowego) poleca obliczyć wartość wyrażenia napisanego po prawej stronie znaku równości i nadać tę wartość zmiennej napisanej po lewej stronie.

Przykład 3.3.1 Przykłady instrukcji arytmetycznych:

L=5	X=-123.567
I=-125×K	Y5=X/Y(J+2)
L(2K+1)=-KI+L(J+6)	Z11=-3.1416×Z
X=STAND I(K+4)	XM=-COS Y(12I+J)
J=-ABS KL	I=ENTIER X129

Przykład 3.3.2 Zmienne występujące po prawej i lewej stronie znaku równości mogą w szczególności być takie same, jak na przykład w instrukcjach

I=I+1 (zwiększenie o 1 aktualnej wartości zmiennej I),

Y=Y×Y (podniesienie aktualnej wartości zmiennej Y do kwadratu),

Y=Y+Y (dwa sposoby podwojenia

Y=2×Y aktualnej wartości zmiennej Y).

Przykład 3.3.3 Przypuśćmy, że należy obliczyć wartość wyrażenia arytmetycznego $\sqrt{x^2+y^2+1}$. Jeżeli zmienne X,Y oznaczają w programie odpowiednio zmienne x,y, to po wykonaniu ciągu instrukcji

```
U=X×X
V=Y×Y
Z=U+V
Z1=Z+1
Z2=SQRT Z1
```

aktualna wartość zmiennej Z2 równa jest wartości podanego wyrażenia arytmetycznego.

Przykład 3.3.4 W poprzednim przykładzie obliczenia są wykonywane w ten sposób, że po ich zakończeniu aktualne wartości kolejno używanych zmiennych równe są kolejnym wynikom pośrednim, tzn. odpowiednio x^2, y^2, x^2+y^2, x^2+y^2+1 i wreszcie $\sqrt{x^2+y^2+1}$. Taki sposób wykonywania obliczeń jest nieoszczędny, ponieważ wymaga użycia tylu zmiennych (tj. tylu komórek pamięci maszyny cyfrowej) ile jest wielkości pośrednich, choć nie wszystkie z nich są potrzebne. Dlatego zaleca się wykonać żądane obliczenia np. za pomocą ciągu instrukcji

```
Z=X×X
Z1=Y×Y
Z=Z+Z1
Z=Z+1
Z=SQRT Z
```

w których występują tylko cztery zmienne, a które dają ten sam rezultat (teraz aktualna wartość zmiennej Z jest równa wartości podanego wyrażenia arytmetycznego).

Jeżeli po wykonaniu obliczeń pierwotne wartości zmiennych X i Y nie są potrzebne, to te same obliczenia można wykonać nie używając zmiennych pomocniczych:

```
X=X×X
X=Y×Y
X=X+Y
X=X+1
X=SQRT X
```

Po wykonaniu tych instrukcji $\sqrt{x^2+y^2+1}$ jest aktualną wartością zmiennej X; jej początkowa wartość została zniszczona.

4. OPISY I INSTRUKCJA START

4.1 Etykiet.

Dla wyróżnienia instrukcji wskazanych w instrukcjach sterujących (§7) i na taśmach z danymi (§5.1) stosuje się etykiety. Etykieta jest liczbą całkowitą bez znaku +. Etykiety naturalne umieszcza się przed instrukcjami, w tym samym wierszu, oddzielając je od instrukcji dwukropkiem. Dowolną instrukcję można opatrzyć tylko jedną etykietą naturalną. Nie można stawiać jednakowych etykiet przed różnymi instrukcjami. Etykieta zerowa 0 użyta w instrukcji sterującej (§7) oznacza instrukcj następującą po tamtej.

Przykład 4.1.1 Instrukcję arytmetyczną I=I+1 opatrzoną etykietą 15 pisze się tak:

15:I=I+1

4.2 Część wstępna programu

Na początku programu umieszcza się w dowolnym porządku tzw. opisy zmiennych i etykiet występujących w tym programie Te opisy zaczynają się odpowiednio od słów INTEGER, REAL, LABEL. Po ostatnim opisie, a przed pierwszą instrukcją prog mu umieszcza się w oddzielnym wierszu słowo BEGIN.

4.3 Opis zmiennych całkowitych

Po słowie INTEGER i pojedynczym odstępie wymienia się w dowolnym porządku nazwy wszystkich zmiennych prostych i tablic typu całkowitego. Dodatkowo, po każdej nazwie tabl cy należy podać maksymalną wartość wskaźnika, używaną w tej tablicy, bez znaku +.

Przykład 4.3.1 Jeżeli w programie są używane tylko następujące zmienne całkowite:

I,LO,L1,...,L25,J,K2,

to opis zmiennych i tablic całkowitych powinien mieć postać

INTEGER IJL25K2

Jeżeli w programie występuje np. zmienna J ze wskaźnikiem I, to należy w opisie zmiennych po literze J napisać największą możliwą w programie wartość zmiennej I jako wskaźnika. Jeżeli w programie nie występuje żadna zmienna całkowita, to opis zmiennych i tablic typu całkowitego pomija się.

4.4 Opis zmiennych zmiennoprzecinkowych

Opis zmiennych prostych i tablic typu zmiennoprzecinkowego różni się od opisu z §4.3 tylko tym, że zaczyna się od słowa REAL. Pomija się go, jeżeli nie ma w programie żadnej zmiennej zmiennoprzecinkowej.

Jeżeli któryś z opisów zmiennych i tablic nie mieści się w jednym wierszu, to piszemy go w dwu lub więcej wierszach zaczynając każdy właściwym ze słów REAL lub INTEGER.

Przykład 4.4.1 Jeżeli w programie wykonuje się działania na dwu macierzach kwadratowych, których stopnie nie przekraczają 20, dwóch macierzach prostokątnych o co najwyżej 10 wierszach i 20 kolumnach i macierze te mają być umieszczone w tablicach typu zmiennoprzecinkowego o nazwach odpowiednio A,B,C,D, a ponadto w tablicach zmiennoprzecinkowych T,U,X,Y,Z trzeba zapamiętać składowe 5 wektorów wymiaru co najwyżej 10, to opis tablic typu zmiennoprzecinkowego może mieć postać:

REAL A399B399C199D199
REAL T9U9X9Y9Z9

4.5 Opis etykiet

Po słowie LABEI i pojedynczym odstępie pisze się maksymalną wartość etykiety użytej w programie. Tej części opisów nie wolno opuszczać, ponieważ w każdym programie musi wystąpić conajmniej jedna etykieta (§4.6).

Przykład 4.5.1 Jeżeli program zawiera opis etykiet

LABEL 50

to etykietami mogą być w tym programie dowolne z liczb 1,2,.. ...,50.

4.6 Instrukcja START

Ostatnią instrukcją w programie (por. przykład 2.7.1) jest instrukcja

START n

(gdzie n>0). Instrukcja ta zawiera etykietę tej instrukcji, od której należy zacząć wykonywanie programu.

5. INSTRUKCJE CZYTANIA

5.1 Czytanie liczb

Większość programów wykonuje obliczenia na danych, które uprzednio wydziurkowano na taśmie papierowej. Liczby dane pisze się w ten sam sposób, jak stałe w programach, z tą tylko różnicą, że na taśmach z danymi liczby dodatnie można poprzedzać znakiem +, a równe jedności mantysy liczb zmiennoprzecinkowych (§2.3, (c)) można opuszczać.

Przykład 5.1.1 Liczbę 0.00001 na taśmie z danymi można napisać w postaci '-5 lub +'-5 .

Symbolem końca liczby na taśmie z danymi są dwa kolejne

odstępy albo przecinek, albo znak zmiany wiersza. Jeżeli nie została przeczytana żadna cyfra, to każdy z symboli końca liczby jest pomijany przez maszynę. Znaki ? (po symbolu końca liczby), znak liter, znak cyfr, znak powrotu karetki i blank są na taśmach z danymi pomijane.

Przykład 5.1.2 Liczby na taśmie z danymi mogą być wydziurkowane w postaci, która w druku jest następująca:

0.78539816,15,40.05

125,'-6,20,

Te same liczby mogą być napisane w inny sposób:

0.78539816 15 40.05 125 '-6 20,

Do wprowadzania do pamięci maszyny liczb wydziurkowanych na taśmie służą instrukcje

READ I lub READ X

Wykonanie takiej instrukcji powoduje przeczytanie najbliższej liczby z taśmy założonej do czytnika maszyny i nadanie zmiennej, której symbol występuje po słowie READ, wartości równej przeczytanej liczbie.

Przykład 5.1.3 Przypuśćmy, że do czytnika maszyny założono od początku taśmę, na której wydziurkowano liczby tak, jak w przykładzie 5.1.2 (pierwsza z tych liczb jest w przybliżeniu równa $\pi/4$). Po wykonaniu przez maszynę instrukcji

```
READ X
X=SIN X
X=X×X
READ Y
Y=Y×X
```

aktualna wartość zmiennej Y będzie równa 7.5 .

Instrukcja READ pozwala także sterować programem z zewnątrz, z taśmy z danymi. Jeżeli zamiast liczby zostanie przeczytana etykieta, tj. liczba naturalna bez znaku + zakończona dwukropkiem (zamiast przecinka, dwu odstępów lub znaku zmiany wiersza albo przed tymi znakami), to wartość zmiennej podanej w instrukcji READ nie zostanie zmieniona, a maszyna przejdzie do wykonania instrukcji z tą etykietą. Jeżeli zamiast liczby zostaną przeczytane dwa dwukropki, to maszyna zatrzyma się.

5.2 Czytanie znaków

Wykonanie instrukcji

INPUT I

powoduje przeczytanie kolejnego znaku z taśmy i nadanie zmiennej I wartości równej kodowi dalekopisowemu przeczytanego znaku. Kody dalekopisowe poszczególnych znaków przytoczone są w poniższej tablicy.

ZNAK LITERY	ZNAK CYFRY	KOD DALEKOPISOWY	ZNAK LITERY	ZNAK CYFRY	KOD DALEKOPISOWY
blank		0	T	5	16
E	3	1	Z	+	17
zmiana wiersza (lf)		2	L	)	18
A	-	3	W	2	19
odstęp (sp)		4	H	☑	20
S	'	5	Y	6	21
I	8	6	P	0	22
U	7	7	Q	1	23
powrót karetki (cr)		8	O	9	24
D	≠	9	B	?	25
R	4	10	G	⊟	26
J	ℛ	11	znak cyfr (fs)		27
N	,	12	M	.	28
F	□	13	X	/	29
C	:	14	V	=	30
K	(	15	znak liter (ls)		31

Znak liter (ls) ustawia dalekopis na drukowanie znaków z kolumny LITERY, znak cyfr (fs) - znaków z kolumny CYFRY

6. INSTRUKCJE DRUKOWANIA I DZIURKOWANIA

6.1 Uwagi ogólne

Maszyna cyfrowa ODRA 1003 może drukować wyniki za pomocą dalekopisu, bądź dziurkować je na taśmie papierowej przy pomocy szybkiego perforatora. W autokodzie MOST 1 tym dwu możliwościom maszyny odpowiadają instrukcje zaczynające się odpowiednio od słowa PRINT (drukowanie na dalekopisie) i PUNCH (dziurkowanie na taśmie). Przy wyborze jednej z dwóch postaci wyników mogą być pomocne poniższe informacje i uwagi.

Szybkość drukowania na dalekopisie nie przekracza 7 do 10 znaków na sekundę; szybkość dziurkowania taśmy przez perforator wynosi do 150 znaków na sekundę. Wydawanie wyników na dalekopisie pozwala łatwo obserwować przebieg obliczeń, ponieważ daje wyniki w bezpośrednio czytelnej postaci. Kopię drukowanych wyników można otrzymać wydziurkowaną na taśmie; z tej taśmy można później, w razie potrzeby, wydrukować wyniki jeszcze raz. Jeżeli jednak wyników jest dużo w porównaniu z liczbą działań potrzebnych do ich uzyskania, to drukowanie wyników na dalekopisie będzie zwykle powodować znaczne straty czasu maszyny, której szybkość w tych warunkach będzie ograniczona szybkością działania dalekopisu.

Dziurkowanie wyników daje tylko taśmę zawierającą dokładnie te same znaki, które przy stosowaniu poprzednio opisanego sposobu byłyby wydrukowane na dalekopisie maszyny. Dziurkowanie jest o wiele szybsze (licząc tylko czas maszyny) i pewniejsze ze względu na znacznie większą - w porównaniu z dalekopisem - niezawodność pracy perforatora. Wyniki z taśmy drukuje się później za pomocą zwykłego dalekopisu z nadajnikiem,

niezależnego od maszyny cyfrowej.

Wykonanie każdej z wymienionych w §§6.2 i 6.3 instrukcji powoduje

(a) wydrukowanie (wydziurkowanie) znaku -, jeżeli drukowana (dziurkowana) liczba jest ujemna lub odstępu w przypadku przeciwnym,

(b) wydrukowanie (wydziurkowanie) kolejnych cyfr (ewentualnie z kropką lub apostrofem) aktualnej wartości zmiennej, której symbol występuje po słowie PRINT lub PUNCH w postaci określonej w instrukcji. Początkowe nieznaczące zera zastępuje się odstępami. Znak liczby umieszcza się bezpośrednio przed pierwszą cyfrą znaczącą.

(c) wydrukowanie (wydziurkowanie) dwu odstępów.

Taśmy zawierające tylko liczby wydziurkowane przez maszynę można stosować jako dane do obliczeń bez zmiany tych taśm.

6.2 Drukowanie i dziurkowanie liczb całkowitych

Wykonanie instrukcji

PRINT I,n lub PUNCH I,n

(gdzie $1 \leqslant n \leqslant 12$) powoduje wydrukowanie (wydziurkowanie) aktualnej wartości zmiennej I z n cyframi. Drukuje się (dziurkuje) n+3 znaki dalekopisowe. Jeżeli liczba ma więcej niż n cyfr, to maszyna drukuje (dziurkuje) tę liczbę w nowym wierszu zgodnie z instrukcją PRINT I,12 (PUNCH I,12) dodając po symbolu końca tej liczby znak ?.

Instrukcje

PRINT I lub PUNCH I

drukują (dziurkują) wartość zmiennej I z liczbą cyfr określoną przez poprzednią instrukcję drukowania lub dziurkowania

liczby całkowitej. Jeżeli poprzednio nie wydrukowano (wydziurkowano) żadnej liczby całkowitej, to liczba cyfr może być przypadkowa.

Przykład 6.2.1 Niech w tym przykładzie (i wszystkich dalszych podobnych przykładach) znak _ oznacza odstęp. Jeżeli aktualna wartość zmiennej I jest równa 525 i maszyna wykona instrukcję

PRINT I,6

to zostaną wydrukowane znaki

____525__

Jeżeli zmienna I ma wartość -1256, to wykonanie tej samej instrukcji spowoduje wydrukowanie znakow

__-1256__

Gdyby jednak wartość zmiennej I była równa 1234512, to wykonanie tej samej instrukcji spowodowałoby wydrukowanie znaków

cr lf cr_____1234512__?

6.3 Drukowanie i dziurkowanie liczb zmiennoprzecinkowych

Instrukcja

PRINT X,m.n lub PUNCH X,m.n

(gdzie $1 \leq m+n \leq 9$) drukuje (dziurkuje) aktualną wartość zmiennej zmiennoprzecinkowej X z m cyframi przed kropką i n cyframi po kropce dziesiętnej. Jeżeli n=0, to zamiast kropki dziesiętnej drukuje się (dziurkuje) odstęp. W każdym przypadku maszyna drukuje (dziurkuje) m+n+4 znaki dalekopisowe.

Przykład 6.3.1 Jeżeli aktualna wartość zmiennej X jest równa 5.1475, to wykonanie instrukcji

PRINT X,3.2

spowoduje wydrukowanie znaków

___5.15__

a wykonanie instrukcji

PRINT X,3.0

-wydrukowanie znaków

___5___

Instrukcja

PRINT X,n' lub PUNCH X,n'

(gdzie $1 \leqslant n \leqslant 9$) drukuje (dziurkuje) aktualną wartość zmiennej zmiennoprzecinkowej X w postaci znormalizowanej, z n cyframi znaczącymi. Postacią znormalizowaną liczby nazywa się postać omówioną w §2.3, (c), w której mantysa M spełnia warunek $0.1 \leqslant M < 1$. Maszyna drukuje (dziurkuje) w tym przypadku znak liczby, kropkę, n cyfr mantysy liczby, apostrof, znak cechy C (minus lub odstęp), cyfrę dziesiątek cechy (może być zero), cyfrę jednostek i zwykły symbol końca liczby; ogółem maszyna drukuje (dziurkuje) w tym przypadku n+8 znaków dalekopisowych.

Przykład 6.3.2 Jeżeli aktualna wartość zmiennej zmiennoprzecinkowej A5 jest równa -9.81, to wykonanie instrukcji

PUNCH A5,4'

spowoduje wydziurkowanie znaków

-.9810'_01__

a jeżeli wartość zmiennej A5 równa jest 0.33443, to wykonanie tej instrukcji spowoduje wydziurkowanie znaków

_.3344'_00__

Instrukcje

PRINT X lub PUNCH X

w których nie określono postaci drukowania (dziurkowania) drukują (dziurkują) liczbę zmiennoprzecinkową w tej postaci, jak poprzednio wykonana instrukcja drukowania lub dziurkowania liczby zmiennoprzecinkowej. Jeżeli nie była dotąd drukowana (dziurkowana) żadna liczba zmiennoprzecinkowa, to postać wydrukowanej liczby może być przypadkowa.

Jeżeli w czasie wykonywania jednej z instrukcji PRINT X,m.n lub PUNCH X,m.n albo PRINT X lub PUNCH X wartość zmiennej X nie może być poprawnie wydrukowana (wydziurkowana) w żądany sposób, to maszyna drukuje (dziurkuje) tę liczbę w nowym wierszu zgodnie z instrukcją PRINT X,9' (PUNCH X,9') dodając po symbolu końca tej liczby znak ?.

Przykład 6.3.3 Jeżeli wartość zmiennej zmiennoprzecinkowej B14 jest równa 9874.19, to wykonanie instrukcji

PRINT B14,3.2

spowoduje wydrukowanie znaków

cr lf cr _.987419000'_04__?

Wszystkie znaki drukowane (dziurkowane) w czasie wykonywania dowolnej z instrukcji opisanych w §§ 6.2 i 6.3 są znakami z kolumny CYFRY (§5.2) jednak pod warunkiem, że nie zostanie to świadomie lub przypadkowo zmienione przez programistę za pomocą instrukcji z §6.5 i że w czasie drukowania wyników nie zostanie ręcznie zmienione ustawienie dalekopisu.

6.4 Drukowanie i dziurkowanie tekstu

Dla uzyskania większej czytelności wyników dostarczonych przez maszynę, można je opatrywać tytułami, objaśnieniami

użytych symboli i :p. Do tego celu służy instrukcja

PRINT ₰Tekst₰ lub PUNCH ₰Tekst₰

gdzie symbol Tekst oznacza dowolny układ znaków dalekopisowych różnych od znaku ₰. Wykonanie tej instrukcji powoduje wydrukowanie (wydziurkowanie) tego układu znaków oraz znaku cyfr fs (por. ostatni akapit §6.3).

Na formularzach programów pomiędzy znakami ₰ pisze się kolejne znaki do drukowania (dziurkowania), zaznaczając także wyraźnie odpowiednimi symbolami znaki decydujące o układzie graficznym tekstu, mianowicie odstępy (symbol sp), znaki powrotu karetki (cr) i znaki zmiany wiersza (lf). Dla zaznaczenia, że pewien znak ma być w tytule powtórzony wielokrotnie stosuje się symbole typu sp^n, lf^n i podobne. Jeżeli np. w tytule mają być umieszczone znaki cr lf|lf sp sp sp, to na formularzu programu piszemy we właściwym miejscu symbole cr lf^2 sp^3. Znaku liter (ls) i znaku cyfr (fs) nie uwidacznia się w ten sposób.

Przykład 6.4.1 Jeżeli aktualna wartość zmiennej Z jest równa -0.25, to wykonanie instrukcji, które na formularzu programu mają postać

```
PRINT ₰cr lf cr ALFA sp =₰
PRINT Z,3.2
```

spowoduje wydrukowanie od nowego wiersza znaków

ALFA_=_-10.25__

6.5 Drukowanie i dziurkowanie znaków

Pojedyncze znaki o danym kodzie dalekopisowym można drukować (dziurkować) za pomocą instrukcji

(a) PRINTOU n lub PUNCHOUT n

(b) PRINTOUT I lub PUNCHOUT I

Wykonanie instrukcji (a) powoduje wydrukowanie (wydziurkowanie) znaku o kodzie dalekopisowym n, a wykonanie instrukcji (b) - wydrukowanie (wydziurkowanie) znaku o kodzie równym aktualnej wartości zmiennej I. Ściślej, jeżeli n > 31 lub I jest ujemne lub większe od 31, to maszyna drukuje (dziurkuje) znak o kodzie dalekopisowym równym nieujemnej reszcie z dzielenia n lub I przez 32.

Przykład 6.5.1 Instrukcja PRINTOUT 31 ustawia dalekopis na drukowanie znaków z kolumny LITERY (§5.2). Ten sam skutek miałyby instrukcje

```
I=-1
PRINTOUT I
```

6.6 Drukowanie i dziurkowanie znaków układu graficznego

Do drukowania wyników obliczeń w czytelnym układzie graficznym używa się instrukcji

(a) PRINTSPACE n lub PUNCHSPACE n

PRINTSPACE I lub PUNCHSPACE I

(b) PRINTLINE n lub PUNCHLINE n

PRINTLINE I lub PUNCHLINE I

Wykonanie instrukcji (a) powoduje wydrukowanie (wydziurkowanie n albo I znaków odstępu (sp), a wykonanie instrukcji (b) - wydrukowanie (wydziurkowanie) znaku powrotu karetki (cr), n albo I znaków zmiany wiersza (lf) oraz jeszcze jednego (ze względów technicznych) znaku powrotu karetki.

Jeżeli n=0 lub I=0, to maszyna nie drukuje (dziurkuje)

żadnego znaku. Jeżeli I < 0, to maszyna nieprzerwanie drukuje przez kilkanaście minut (lub dziurkuje przez kilka minut) znaki sp w przypadku instrukcji (a) i znaki lf w przypadku instrukcji wymienionej w (b).

Przykład 6.6.1 Aby wydrukować w nowym wierszu aktualne wartości zmiennych X1 i X2, a w następnym wierszu - wartości zmiennych Y1 i Y2, wszystkie w postaci znormalizowanej z sześcioma cyframi mantysy, należy w programie umieścić instrukcje

```
PRINTLINE 1
PRINT X1,6'
PRINT X2
PRINTLINE 1
PRINT Y1
PRINT Y2
```

7. INSTRUKCJE STERUJĄCE

7.1 Instrukcje zatrzymania maszyny

Wykonanie instrukcji

STOP lub STOP n

powoduje zatrzymanie maszyny. W pierwszym przypadku ponowne uruchomienie maszyny (tj. naciśnięcie przycisku START) spowoduje ponowne wykonanie instrukcji STOP. W drugim przypadku, po ponownym uruchomieniu maszyna wykona instrukcję z etykietą n i dalsze instrukcje programu. Jeżeli n=0, to zgodnie z tym, co powiedziano w §4.1, maszyna przechodzi do wykonania następnej instrukcji programu.

Przykład 7.1.1 Jeżeli w pewnym etapie obliczeń potrzebne jest, aby maszyna zatrzymała się np. w celu umożliwienia operatorowi założenia nowej taśmy z danymi, albo ustawienia pew-

nej informacji na pulpicie sterowania, to należy we właściwym miejscu programu umieścić instrukcję

STOP 0

jeżeli po ponownym uruchomieniu maszyna ma wykonać następną instrukcję programu, i instrukcję

STOP 25

jeżeli po ponownym uruchomieniu maszyna ma wykonać instrukcję z etykietą 25.

Oprócz tych dwu programowych możliwości zatrzymywania maszyny istnieją jeszcze inne, z których można korzystać z zewnątrz maszyny i niezależnie od programu. W szczególności można zatrzymać maszynę

1° za pomocą instrukcji READ, jeżeli z taśmy z danymi zostaną przeczytane dwa kolejne dwukropki (§5.1) ;

2° przez naciśnięcie przycisku STOP na pulpicie sterowania maszyny.

W tym ostatnim przypadku na ogół trudno jest stwierdzić, w którym miejscu obliczenia zostały zatrzymane.

7.2 Instrukcja skoku bezwarunkowego.

Instrukcja

GO TO n lub GO TO I

poleca przejść do wykonania instrukcji opatrzonej etykietą n

lub - odpowiednio - równą aktualnej wartości zmiennej I; zmienna I może być tutaj tylko zmienną prostą (bez wskaźnika).

Jeżeli n=0, to maszyna przechodzi do instrukcji następującej po instrukcji skoku. Jeżeli I=0, to maszyna zatrzymuje się, a po ponownym jej uruchomieniu wykonuje program od początku.

Jeżeli n lub I jest dodatnie, nie przekracza wartości dopuszczalnej w danym programie (to znaczy nie jest większe od liczby wymienionej w opisie etykiet), ale w programie nie ma instrukcji opatrzonej tą etykietą, to instrukcja skoku bezwarunkowego jest interpretowana dokładnie tak samo, jak instrukcja STOP.

Jeżeli wartość zmiennej I jest ujemna, to wykonanie instrukcji GO TO I może popsuć program.

Te same uwagi stosuje się do wszystkich instrukcji sterujących omówionych dalej.

Przykład 7.2.1 Przypuśćmy, że pewien program zawiera trzy warianty, według których mogą być wykonywane obliczenia. Jeżeli na taśmie z danymi informację o numerze wariantu programu podano w postaci liczby całkowitej 1, 2 lub 3 i wariant 1 zaczyna się od instrukcji z etykietą 15, wariant 2 - od instrukcji z etykietą 19, a wariant 3 - od instrukcji z etykietą 55, to właściwy wariant obliczeń można wybrać za pomocą instrukcji:

```
READ I
GO TO I
1:GO TO 15
2:GO TO 19
3:GO TO 55
```

Przykład 7.2. Omówimy prosty przykład programu, w którym użyjemy dotąd omówionych instrukcji. Przypuśćmy, że dla danych wartości a i b trzeba obliczyć wartości wyrażeń $\sqrt{a^2+b^2+1}$ oraz $-\frac{1}{2}\exp\sqrt{a^2+b^2+1}$. Zakłada się, że po wykonaniu obliczeń i wydrukowaniu wyników dla pewnej pary a i b maszyna wykonuje obliczenia dla następnej pary wydziurkowanej na taśmie. Po ostatniej parze na taśmie wydziurkowano znaki :: . Wyniki należy wydrukować na dalekopisie w postaci tablicy pod tytułem

NUMER PARY	WARTOŚĆ A	WARTOŚĆ B	WARTOŚĆ WYRAŻENIA 1	WARTOŚĆ WYRAŻENIA 2

Numery par mają być wydrukowane jako liczby trzycyfrowe, a pozostałe liczby - w postaci zmiennoprzecinkowej z sześcioma cyframi znaczącymi. To zadanie wykonuje program, który na formularzu ma następującą postać

```
INTEGER I
REAL AB
LABEL 2
BEGIN
1:I=0
PRINT ßcr 1f^4 cr NUMER sp^3 WARTOŚĆ sp^7 WARTOŚĆsp^7 WARTOŚĆ sp^7
WARTOŚĆ cr 1f cr sp PARY sp^7 A sp^13 B sp^8 WYRAŻENIA sp 1 sp^3
WYRAŻENIA sp 2 cr 1f^2 crß
2:I=I+1
READ A
PRINTLINE 1
PRINT I,3
PRINT A,6'
A=A×A
READ B
PRINT B
B=B×B
A=A+B
A=A+1
```

```
A=SQRT A
PRINT A
A=EXP A
A=-.5*A
PRINT A
GO TO 2
START 1
```

Program został ułożony w taki sposób, aby używał możliwie mało zmiennych (por. przykład 3.3.4). Wyniki obliczone i wydrukowane przez ten program dla danych wydziurkowanych na taśmie w postaci jak niżej

.1,.2 .2,.3 .3,.4,::

mają postać, którą przytaczamy

NUMER PARY	WARTOŚĆ A	WARTOŚĆ B	WARTOŚĆ WYRAŻENIA 1	WARTOŚĆ WYRAŻENIA 2
1	.100000' 00	.200000' 00	.102470' 01	-.139312' 01
2	.200000' 00	.300000' 00	.106301' 01	-.144754' 01
3	.300000' 00	.400000' 00	.111803' 01	-.152942' 01

Właściwe rozmieszczenie tytułów, a następnie wydrukowanie wyników we właściwych miejscach osiągamy wykonując przed ukończeniem ostatecznej postaci programu dokładną makietę wyników z uwidocznieniem ilości znaków, które będą wydrukowane (wydziurkowane) przez maszynę (p. §§ 6.2 i 6.3). Przy przygotowywaniu makiety pamiętać należy, że liczba znaków, które mieszczą się w jednym wierszu dalekopisu wynosi 69; długość arkusza praktycznie nie jest ograniczona, w związku z czym można projektować dowolny podział wyników na stronice. Znaki bl, fs oraz ls nie powodują przesuwania karetki dalekopisu, nie należy więc ich wliczać do ogólnej liczby znaków w wierszu.

7.3 Instrukcja skoku zewnętrznego

Instrukcja

GO TO I J IF BUTTON n

(gdzie $9 \leq n \leq 21$) jest poleceniem przejścia do wykonania instrukcji z etykietą I, jeżeli na pulpicie sterowania nie jest wciśnięty n-ty przycisk klawiatury akumulatora, a przejścia do wykonania instrukcji z etykietą J, jeżeli ten przycisk jest wciśnięty. Zmienne I oraz J mogą być tylko zmiennymi prostymi. Każdą z nich można zastąpić nieujemną liczbą całkowitą, przy czym zero oznacza następną instrukcję.

7.4 Instrukcja skoku warunkowego

Wykonanie instrukcji

GO TO I J K IF L=A lub GO TO I J K IF X=B

gdzie napis L=A lub X=B ma taką postać, jak dowolna instrukcja arytmetyczna (§3.3), powoduje obliczenie wartości W różnicy L-A lub X-B i przejście do wykonania instrukcji z etykietą

I, jeżeli $W < 0$,

J, jeżeli $W = 0$,

K, jezeli $W > 0$.

Zmienne I J K mogą być tutaj tylko zmiennymi prostymi. Każdą z nich można zastąpić nieujemną liczbą całkowitą, przy czym zero oznacza następną instrukcję.

Przykład 7.4.1 Przypuśćmy, że jeżeli $X < Y$, to maszyna ma przejść do wykonania instrukcji z etykietą 12, a w przeciwnym przypadku - do wykonania następnej instrukcji. Osiąga się to umieszczając w odpowiednim miejscu programu instrukcję

GO TO 12 0 0 IF X=Y

Przykład 7.4.1 Przypuśćmy, że trzeba przejść do wykonania instrukcji z etykietą 5, jeżeli są jednocześnie spełnione warunki X=ABS Y i U=Z, a w przeciwnym przypadku ma być wykonana instrukcja z etykietą 11. Służą do tego instrukcje

GO TO 11 0 11 IF X=ABS Y

GO TO 11 5 11 IF U=Z

7.5 Skrócona instrukcja skoku warunkowego

Po dowolnej instrukcji arytmetycznej (§3.3), instrukcji READ (§5.1) lub INPUT (§5.2) nadającej wartość jakiejś zmiennej D, można użyć instrukcji skoku warunkowego w skróconej postaci

GO TO m n p

Jest ona równoważna instrukcji

GO TO m n p IF D=0

Skróconej instrukcji skoku warunkowego nie można poprzedzać etykietą.

7.6 Przykłady

Przytoczony niżej przykład, podobnie jak i dalsze, jest fragmentem większego programu, nie zawiera więc opisu zmiennych i etykiet oraz zwykłego zakończenia programu. W tym przypadku - i innych podobnych, gdzie nie podano opisu zmiennych - należy przyjąć, że typy zmiennych występujących w przykładzie są zgodne z §2.5.

Przykład 7.6.1 Przypuśćmy, że do czytnika maszyny założona jest taśma, na której jest pewna ilość znaków bl, po czym następuje znak = i tekst zakończony znakiem bl; należy napi-

sać fragment programu, który wydrukuje ten tekst na dalekopisie, ale tylko wtedy, gdy przed przystąpieniem do drukowania na klawiaturze akumulatora jest wciśnięty przycisk Nr 21; w przypadku przeciwnym taśma powinna być przesunięta aż do znaku bl kończącego tekst. Ten fragment programu może mieć następującą postać:

```
1:INPUT I
GO TO 1 0 1 IF I=30
GO TO 3 2 IF BUTTON 21
2:INPUT I
GO TO 0 10 0
PRINTOUT I
GO TO 2
3:INPUT I
GO TO 0 0 3
10: dalszy ciąg programu
```

Przykład 7.6.2 Przypuśćmy, że w pewnym etapie obliczeń trzeba obliczyć wartość wielomianu

$$u_0x^n + u_1x^{n-1} + \ldots + u_{n-1}x + u_n \quad (n > 0),$$

którego stopień jest aktualną wartością zmiennej N, a współczynniki zapamiętane są w tabliczy U w taki sposób, że $U=u_0$, $U1=u_1, \ldots, UN=u_n$. Po obliczeniach aktualna wartość zmiennej Z ma być równa wartości wielomianu w punkcie równym wartości zmiennej X. Przytoczony przykład działa według schematu Hornera, tzn. oblicza się wartość wielomianu wykonując kolejno zaznaczone niżej działania w kolejności określonej nawiasami:

$$(\ldots((u_0x + u_1)x + u_2)x + \ldots + u_{n-1})x + u_n$$

W autokodzie ten fragment obliczeń ma taką postać:

```
I=0
Z=U
1:I=I+1
Z=Z×X
Z=Z+UI
GO TO 1 0 0 IF I=N
```

Przykład 7.6.3 Podamy fragment programu znajdujący numer tego współczynnika wielomianu, który ma największy moduł. Po wykonaniu tego fragmentu K jest szukanym numerem, a Y=|UK| :

```
Y=0
I=N
5:X=ABS UI
GO TO 0 0 6 IF Y=X
Y=X
K=I
6:I=I-1
GO TO 0 5 5
```

Przykład 7.6.4 Pokażemy teraz nieco trudniejszy fragment programu, który uporządkuje liczby $u_0, u_1, \ldots, u_n$ od najmniejszej do największej i przestawione pozostawi w tej samej tablicy. W opisanej niżej metodzie porządkowania należy wykonać $\frac{1}{2}n(n+1)$ porównań i ewentualnie przestawień liczb. Najpierw porównujemy u_0 i u_1. Jeżeli $u_0 > u_1$, to przestawiamy te dwie liczby, tzn. zamieniamy ich oznaczenia. Jeżeli $u_0 > u_2$, to przestawiamy u_0 i u_2. Te same czynności wykonujemy kolejno dla par u_0 i u_3, u_0 i u_4 itd. W następnym cyklu te same czynności wykonujemy dla kolejnych par u_1 i u_2, u_1 i u_3, u_1 i u_4 itd, a potem dla par u_2 i u_3, u_2 i u_4, u_2 i u_5 itd. Fragment programu wykonujący te czynności może mieć postać jak na następnej stronicy:

```
I=0
1:GO TO 0 5 0 IF I=N
J=I+1
4:GO TO 2 2 0 IF J=N
I=I+1
GO TO 1
2:GO TO 3 3 0 IF UI=UJ
Y=UI
UI=UJ
UJ=Y
3:J=J+1
GO TO 4
5: dalszy ciąg programu
```

Zauważmy, że podany fragment programu działa poprawnie także dla N=0. Taka uniwersalność jest często w programach potrzebna.

7.7 Instrukcje sterowania podprogramami

Podprogramem nazywamy każdy ciąg instrukcji zaczynający się od instrukcji opatrzonej etykietą i kończący się instrukcją END. Do tzw. wywołania podprogramu służy instrukcja

PROCEDURE n lub PROCEDURE I

Instrukcja ta poleca przejść do wykonania podprogramu zaczynającego się od instrukcji opatrzonej etykietą n (lub etykietą równą aktualnej wartości zmiennej I). Wartość n lub I powinna być dodatnia.

Instrukcja END poleca powrót do programu głównego lub podprogramu nadrzędnego, to jest wykonanie instrukcji następującej bezpośrednio po tej instrukcji PROCEDURE n lub PROCEDURE I, która spowodowała wejście do podprogramu.

W instrukcji PROCEDURE I zmienna I może być tylko zmienną

prostą (bez wskaźnika).

Jeżeli wejście do podprogramu nastąpiło przez instrukcję PROCEDURE n lub PROCEDURE I, to wyjście z tego podprogramu musi być spowodowane przez instrukcję END, a nie przez jakąś instrukcję skoku (§§ 7.2÷7.5). Jeżeli wejście do podprogramu nastąpi w inny sposób (przez instrukcję skoku lub zgodnie ze zwykłym porządkiem wykonywania programu), to wyjście musi nastąpić przez instrukcję skoku, a nie przez instrukcję END. Niespełnienie tych warunków może spowodować dezorganizację programu.

Liczba jednocześnie wykonywanych podprogramów, to jest liczba kolejno wykonanych instrukcji PROCEDURE n lub PROCEDURE I bez wykonania odpowiedniej instrukcji END, nie może przekroczyć 10.

Przykład 7.7.1 Przypuśćmy, że współczynniki dwu wielomianów

$$W_1(x) = u_0x^p + u_1x^{p-1} + \ldots + u_{p-1}x + u_p \quad (p \leq 20),$$

$$W_2(x) = u_{20}x^q + u_{21}x^{q-1} + \ldots + u_{20+q-1}x + u_{20+q}$$

zapamiętano w tablicy U w taki sposób, że UI=u_i. Należy napisać fragment programu taki, że po jego wykonaniu Z1=W_1(X), Z2=W_2(X). Zastosowanie podprogramu pozwoli nie pisać drukrotnie ciągu instrukcji obliczających wartość wielomianu:

```
I=0                      15:Z2=UI
N=P                      16:I=I+1
PROCEDURE 15             Z2=Z2×X
Z1=Z2                    Z2=Z2+UI
I=20                     GO TO 16 0 0 IF I=N
N=I+Q                    END
PROCEDURE 15
```

Stosowanie podprogramów w przypadkach podobnych do omówionego skraca zwykle program, choć wydłuża jego wykonanie.

8. INSTRUKCJE TYPU "DLA"

8.1 Uwagi ogólne

Często jest wygodna, a nawet konieczna, taka organizacja programu, że niektóre układy instrukcji są wykonywane wielokrotnie dla zmieniającej się w określony sposób wartości ustalonej zmiennej. Takie układy instrukcji będziemy nazywać pętlami. W niektórych wcześniej omówionych przykładach (7.6.2 - - 7.6.4, 7.7.1) pokazano, jak można sterować pętlami posługując się tylko instrukcjami arytmetycznymi i instrukcjami skoku oraz etykietami. Częściej jednak używa się w tym celu instrukcji zaczynających się od słowa FOR (np. FOR X=Y STEP Z UNTIL U) i instrukcji zaczynających się od słowa END (np. END X). Pierwsza z tych instrukcji występuje zawsze na początku pętli, druga - kończy pętlę. Po słowie FOR wymienia się nazwę zmiennej, której wartość należy zmieniać przy kolejnych powtórzeniach pętli, a dalej określa się sposób zmiany tej wartości.

Każdej instrukcji zaczynającej się od FOR odpowiada dokładnie jedna instrukcja zaczynająca się od END.

W §§ 8.2 - 8.4 symbol (Pętla) oznacza dowolny układ instrukcji nie zmieniający wartości żadnej ze zmiennych wymienionych w instrukcji FOR. Ponadto zakłada się, że zmienna po lewej stronie znaku równości w instrukcji FOR nie jest identyczna z żadną ze zmiennych po prawej stronie tego znaku. Przypadki, w których założenia te nie są spełnione, można zbadać posługując się przykładami z §8.5.

Wszystkie zmienne w instrukcji FOR mogą mieć tylko wskaźniki proste. Każdą ze zmiennych po prawej stronie znaku równości można zastąpić stałą odpowiedniego typu, bez znaku albo ze

znakiem - (minus) dla stałych ujemnych.

8.2 Instrukcja typu "dla" pierwszego rodzaju

Układ instrukcji

(a)
```
FOR I=J STEP K REPEAT L
   (Pętla)
END I
```

lub

(b)
```
FOR X=Y STEP Z REPEAT L
   (Pętla)
END X
```

powoduje w obu przypadkach wykonanie pętli L razy, w przypadku (a) kolejno dla wartości zmiennej I równych

J, J+K, J+2K,..., J+(L-1)K,

a w przypadku (b) kolejno dla wartości zmiennej X równych

Y, Y+Z, Y+2Z,..., Y+(L-1)Z.

Po całkowitym wykonaniu pętli zmienna I (lub X) ma wartość J+L×K (lub Y+L×Z).

Jeżeli wartość L nie jest dodatnia, to pętla nie zostanie wykonana ani razu, a maszyna przejdzie do wykonania instrukcji następującej bezpośrednio po odpowiedniej instrukcji END.

Przykład 8.2.1 Fragment programu obliczający wartość wyrażenia $u_1+ u_2+...+ u_n$ może mieć postać

```
X=0
FOR I=1 STEP 1 REPEAT N
X=X+UI
END I
```

Przykład 8.2.2 Fragment programu z przykładu 7.6.2 obliczający wartość wielomianu według schematu Hornera można teraz

napisać w następujący sposób:

```
Z=U
FOR I=1 STEP 1 REPEAT N
Z=Z×X
Z=Z+UI
END I
```

Zauważmy, że ten wariant schematu Hornera jest nieco krótszy od poprzedniego i że nie ma w nim żadnej etykiety.

Przykład 8.2.3 Przypuśćmy, że dana jest macierz kwadratowa U stopnia n o współczynnikach ponumerowanych jak poniżej:

$$\begin{bmatrix} u_1 & u_2 & \ldots\ldots & u_n \\ u_{n+1} & u_{n+2} & \ldots\ldots & u_{2n} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ u_{(n-1)n+1} & & \ldots\ldots & u_{n\times n} \end{bmatrix}$$

i zapamiętana w tablicy U w taki sposób, że u_i = UI. Dany jest również wektor X o składowych $x_1, x_2, \ldots, x_n$ zapamiętany w tablicy X w taki sposób, że x_i = XI. Należy napisać fragment programu obliczający składowe y_i wektora Y = U X (iloczyn macierzy przez wektor) i umieszczający je w tablicy Y w taki sposób, że y_i = YI. Jeden z możliwych sposobów wykonania tego zadania jest następujący:

```
K=0
FOR J=1 STEP 1 REPEAT N
Y=0
FOR I=1 STEP 1 REPEAT N
K=K+1
Z=UK×XI
Y=Y+Z
END I
YJ=Y
END J
```

8.3 Instrukcja typu "dla" drugiego rodzaju

(a) Typ całkowity

Układ instrukcji

FOR I=J STEP K UNTIL L

(Pętla)

END I

powoduje wykonanie pętli kolejno dla tych wszystkich wartości zmiennej I równych J, J+K, J+2K,..., które należą do przedziału $\langle J,L\rangle$.

Przykład 8.3.1 Pętla poprzedzona instrukcją

FOR I=1 STEP 5 UNTIL 20

będzie wykonana dla wartości zmiennej I równych kolejno 1,6,11,16.

Przykład 8.3.2 Pętla poprzedzona instrukcją

FOR I=1 STEP 2 UNTIL 9

będzie wykonana dla I=1,3,5,7,9.

Jeżeli $\frac{L-J}{K}<0$ (jak np. w instrukcji FOR I=1 STEP 5 UNTIL 0), to pętla nie jest wykonywana i maszyna przechodzi bezpośrednio do wykonania instrukcji napisanej po odpowiedniej instrukcji END I.

W pewnych przypadkach instrukcje "dla" pierwszego i drugiego rodzaju dają te same rezultaty, jak na przykład instrukcje

FOR I=1 STEP 1 REPEAT N

i FOR I=1 STEP 1 UNTIL N

W takich przypadkach wybór rodzaju instrukcji "dla" nie ma istotnego znaczenia.

Przykład 8.3.3 Fragment programu z przykładu 7.6.3 można napisać także w postaci

```
Y=0
FOR I=N STEP -1 UNTIL 0
X=ABS UI
GO TO 0 0 6 IF Y=X
Y=X
K=I
6:END I
```

Tutaj także zauważamy skrócenie tego fragmentu i możliwość pominięcia etykiety na początku pętli.

(b) Typ zmiennoprzecinkowy

Układ instrukcji

```
FOR X=Y STEP Z UNTIL U
  (Pętla)
END X
```

powoduje wykonanie pętli dla tych wszystkich wartości zmiennej X równych kolejno Y, Y+Z, Y+2Z,..., które należą do przedziału $\langle Y, U + \frac{1}{2}|Z| \rangle$, przy czym ostatnią z tych wartości zastępuje się wartością U. Punktu $U + \frac{1}{2}|Z|$ do przedziału nie zalicza się.

Przykład 8.3.4 Pętla poprzedzona instrukcją

```
FOR X=1 STEP 1 UNTIL 5.2
```

będzie wykonana dla wartości zmiennej X równych kolejno 1,2, 3,4,5.2, a pętla poprzedzona instrukcją

```
FOR X=1 STEP 1 UNTIL 3.8
```

- dla wartości zmiennej X równych kolejno 1,2,3,3.8. Zauważmy jeszcze, że instrukcja FOR X=1 STEP 1 UNTIL 3.5 powoduje wykonanie pętli dla wartości X równych kolejno 1,2,3.5, a instrukcja

FOR X=1 STEP 1 UNTIL 3.50001 powoduje wykonanie pętli dla wartości X równych kolejno 1,2,3,3.50001.

Jeżeli $\frac{U-Y}{Z} \leqslant \frac{1}{2}$, to pętla jest wykonywana dokładnie jeden raz dla wartości X=U. Tak więc pętla poprzedzona dowolną z instrukcji

```
FOR X=1 STEP 2 UNTIL 2
FOR X=1 STEP -2 UNTIL 2
```

będzie wykonana dla wartości X=2.

Jeżeli krok pętli (tj. wielkość napisana po słowie STEP) jest w instrukcji "dla" drugiego rodzaju równy zeru, to maszyna zatrzymuje się sygnalizując błąd na pulpicie sterowania.

Po całkowitym wykonaniu instrukcji "dla" drugiego rodzaju zmienna X ma tę wartość, dla której pętla była wykonana po raz ostatni, natomiast zmienna I wartość o K większą.

Przykład 8.3.5 Przypuśćmy, że trzeba obliczyć wartości funkcji sin x dla argumentów zmieniających się od .01 co .31415926 do 31.416 i zapamiętać kolejne wartości w tablicy X poczynając od X0. Fragment programu wykonujący to zadanie może mieć następującą postać

```
I=0
FOR Y=.01 STEP .31415926 UNTIL 31.416
XI=SIN Y
I=I+1
END Y
```

8.4 Instrukcja typu "dla" trzeciego rodzaju

Układ instrukcji

```
FOR I=i,j,k,l,...
   (Pętla)
END I
```

lub

```
FOR X=x,y,z,u,...
  (Pętla)
END X
```

powoduje wykonanie pętli dla wartości zmiennej I (lub X) równych kolejno stałym i,j,k,l,... (lub x,y,z,u,...) napisanym po prawej stronie znaku równości. Stałe te oddziela się przecinkami, a po ostatniej liczbie przecinka nie umieszcza się. Jeżeli instrukcja nie mieści się w jednym wierszu, to piszemy ją w kilku kolejnych wierszach; wiersz, po którym ma nastąpić dalszy ciąg stałych kończymy przecinkiem. Zmiana wiersza, przed którą nie ma przecinka oznacza koniec tej instrukcji. Po ostatnim wykonaniu pętli zmienna I (lub X) ma wartość równą ostatniej stałej.

Przykład 8.4.1 Przypuśćmy, że zmiennym X1,X2,...,X8 chcemy nadać odpowiednio wartości 1,2,5,10,20,50,100,500. Można to wykonać za pomocą instrukcji

```
FOR I=1 STEP 1 UNTIL 8
READ XI
END I
```

umieszczając wymienione liczby we właściwej kolejności na taśmie z danymi. Ten sposób może być niepraktyczny, jeżeli w programie trzeba takie nadawanie wartości wykonać wielokrotnie, ponieważ program będzie wówczas wymagał wielokrotnego zakładania taśmy z danymi do czytnika, bądź też wielokrotnego napisania tego samego układu liczb na taśmie. Można również użyć instrukcji

```
X1=1
X2=2
X3=5
.....
```

i tak dalej aż do instrukcji

X8=500

Ten sposób realizacji zadania byłby jednak nieco uciążliwy i zająłby dużo miejsca w pamięci maszyny w przypadku dużej liczby zmiennych, którym trzeba nadać wartości początkowe. Najlepiej użyć następujących instrukcji:

```
I=1
FOR XI=1,2,5,10,20,50,100,500
I=I+1
END XI
```

Przykład 8.4.2 Przypuśćmy, że pewien fragment programu ma postać

```
PROCEDURE 15
PROCEDURE 1
PROCEDURE 10
PROCEDURE 9
PROCEDURE 15
PROCEDURE 10
```

Można go napisać krócej w postaci

```
FOR I=15,1,10,9,15,10
PROCEDURE I
END I
```

8.5 Uwagi o tłumaczeniu instrukcji typu "dla"

Działanie instrukcji typu "dla" omówiono przy pewnych założeniach (§8.1), które nie są sprawdzane przez translator. W praktyce czasem może być pożyteczne świadome zignorowanie tych założeń pod warunkiem, że programista zdaje sobie sprawę ze wszystkich skutków takiego postępowania. Przewidywanie tych skutków umożliwią podane dalej informacje. Dla uniknięcia zbyt

długich objaśnień podajemy z pewnymi komentarzami fragmenty programów w autokodzie, które są ściśle równoważne pętli poprzedzonej instrukcją typu "dla" i zakończonej odpowiednią instrukcją zaczynającą się od END. We wszystkich podanych przykładach H oznacza pewną zmienną pomocniczą typu całkowitego, a S - pewną zmienną pomocniczą typu zmiennoprzecinkowego. Oczywiście, translator wprowadza osobne zmienne pomocnicze dla każdej z pętli umieszczonych kolejno wewnątrz siebie.

Przykład 8.5.1 Układ instrukcji

```
FOR I=J STEP K REPEAT L
  (Pętla)
END I
```

jest w sensie wykonanych czynności równoważny układowi

```
I=J
H=-L
1:GO TO 0 2 2 IF H=0
   (Pętla)
H=H+1
I=I+K
GO TO 1
2: Instrukcja następująca po END I
```

Wynika stąd m. in., że zmiany wartości zmiennych I,J,K,L wewnątrz pętli nie wpływają na liczbę powtórzeń pętli.

Instrukcja FOR X=Y STEP Z REPEAT L działa zupełnie podobnie.

Przykład 8.5.2 Układ instrukcji

```
FOR I=J STEP K UNTIL L
  (Pętla)
END I
```

działa tak, jak układ

```
I=J
GO TO 2
```

```
1:I=I+K
2:H=L-I
GO TO 0 A 3 IF K=0
H=-H
3:GO TO 4 0 0 IF H=0
   (Pętla)
GO TO 1
4: Instrukcja następująca po END I
```

Litera A oznacza tutaj i w następnym przykładzie, przejście do sygnalizacji na pulpicie sterowania omyłki w przypadku, gdy krok K jest równy zeru.

Przykład 8.5.3 Układ instrukcji

```
FOR X=Y STEP Z UNTIL U
  (Pętla)
END X
```

działa tak samo jak instrukcje

```
X=Y
GO TO 2
1:X=X+Z
2:S=X-U
S=S+S
GO TO 3 A 0 IF Z=0
S=S+Z
GO TO 4
3:S=-S-Z
4:GO TO 0 6 6 IF S=0
PROCEDURE 5
GO TO 1
5: (Pętla)
END
6:X=U
PROCEDURE 5
 Instrukcja następująca po END X
```

Przykład 8.5.4 Układ instrukcji

FOR I=$k_1,k_2,\ldots,k_n$
(Pętla)
END I

gdzie k_i (i=1,2,...,n) są stałymi, działa tak samo, jak instrukcje

H=-n
1:GO TO 0 2 2 IF H=0
H=H+1
I=k_{n+H}
(Pętla)
GO TO 1
2: Instrukcja następująca po END I

Instrukcja FOR X=$y_1,y_2,\ldots,y_n$ działa zupełnie podobnie.

9. OBSERWACJA PRACY PROGRAMU PRZETŁUMACZONEGO

9.1 Warunkowe drukowanie wyników pośrednich

Instrukcja

TEST I lub TEST X

jest uwzględniana przez translator tylko wtedy, gdy w czasie jego pracy na klawiaturze akumulatora maszyny jest wciśnięty przycisk Nr 0; w przeciwnym przypadku translator pomija tę instrukcję i przechodzi do tłumaczenia następnej. Jeżeli ten sam przycisk jest wciśnięty w czasie wykonywania programu przetłumaczonego, to maszyna drukuje kolejno znaki cr lf cr ls, następnie nazwę zmiennej napisanej po słowie TEST (litera i co najwyżej czterocyfrowa liczba całkowita równa aktualnej wartości wskaźnika przy tej zmiennej) i dalej aktualną wartość zmiennej I (lub X) w postaci zgodnej z instrukcją PRINT I,12

(lub - odpowiednio - PRINT X,9'). Po tej liczbie maszyna drukuje znak ✠. Jeżeli przycisk Nr 0 klawiatury akumulatora nie jest wciśnięty w czasie wykonywania programu przetłumaczonego, to instrukcja TEST jest pomijana i maszyna przechodzi do wykonania następnej instrukcji programu.

Przykład 9.1.1 Jeżeli w chwili wykonywania instrukcji

TEST XN

aktualna wartość zmiennej N jest równa 15, a aktualna wartość zmiennej X15 jest równa 9587.521, to po wykonaniu tej instrukcji na dalekopisie maszyny pojawi się w nowym wierszu ciąg znaków

X___15___.958752100'_04__✠

Symbol _ ma takie samo znaczenie, jak w przykładzie 6.2.1.

Jeżeli w chwili wykonywania instrukcji

TEST L(2K-J)

jest K=100 i J=25, zaś L175=256, to dalekopis maszyny wydrukuje w nowym wierszu znaki

L__175_____256__✠

W tym przypadku ten sam skutek będzie miało wykonanie instrukcji TEST L175. Zmienna bez wskaźnika jest w instrukcji TEST traktowana dokładnie tak samo, jak zmienna ze wskaźnikiem stałym równym zero.

Jeżeli wartość wskaźnika przy zmiennej napisanej po słowie TEST jest ujemna, to obie liczby drukowane przez maszynę są fałszywe, co nie jest przez maszynę sygnalizowane.

Ponieważ instrukcja TEST jest uwzględniana warunkowo w programie tłumaczonym, więc po sprawdzeniu programu i stwierdzeniu, że jest on poprawny można przygotować nową taśmę pro-

gramu przetłumaczonego nie zawierającą instrukcji TEST - bez jakiegokolwiek poprawiania programu w autokodzie.

9.2 Warunkowe drukowanie śladu programu

Jeżeli w czasie tłumaczenia programu i następnie w czasie jego wykonywania jest wciśnięty przycisk Nr 1 klawiatury akumulatora, to maszyna przed wykonaniem każdej instrukcji opatrzonej etykietą drukuje w nowym wierszu tę etykietę wraz z dwukropkiem. Wydrukowany w ten sposób wykaz etykiet nazywa się śladem programu, ponieważ umożliwia w pewnym zakresie obserwację biegu programu.

Tłumaczyć program z możliwością drukowania jego śladu warto zawsze wtedy, gdy wiadomo, że program działa źle. Pozwala to dosyć łatwo znaleźć miejsce, w którym powstaje błąd. Uzupełnienie śladu drukowaniem wyników pośrednich przy pomocy instrukcji TEST pozwala bardzo szybko znaleźć błędy w programie pod warunkiem, że występuje w nim dostatecznie dużo instrukcji opatrzonych etykietą. W szczególności można łatwo w ten sposób sprawdzić poprawność działania pętli, jeżeli instrukcja END kończąca pętlę jest opatrzona etykietą.

Program przetłumaczony z uwzględnieniem możliwości drukowania jego śladu jest jednak dłuższy od programu przetłumaczonego w zwykły sposób i dłużej od niego wykonywany nawet wtedy, gdy w czasie wykonywania programu nie korzysta się z tej możliwości. Z tego powodu możliwość drukowania śladu programu realizuje się tylko w przypadkach uzasadnionych. To samo dotyczy tłumaczenia i wykonywania instrukcji TEST.

10. BLOKI W KODZIE WEWNĘTRZNYM MASZYNY

10.1 Uwagi ogólne

Do programu napisanego w autokodzie można włączać bloki napisane w kodzie wewnętrznym maszyny (zob. [1]). Każdy taki blok należy poprzedzić nawiasem otwierającym i zakończyć nawiasem zamykającym. Każdy z nawiasów i każde słowo bloku umieszcza się w nowym wierszu. Żadnego ze słów bloku nie można poprzedzić etykietą. Etykietę można natomiast umieścić przed nawiasem otwierającym; taka etykieta odnosi się do pierwszego słowa bloku.

Wejście do bloku w kodzie wewnętrznym może nastąpić w zwykły sposób, przy czym pierwszy rozkaz napisany po nawiasie otwierającym jest wykonywany bezpośrednio po instrukcji poprzedzającej ten nawias, lub przez dowolną instrukcję skoku. Podobnie wyjście z bloku w kodzie maszyny może nastąpić przez wykonanie dowolnego rozkazu skoku do dowolnej instrukcji programu opatrzonej etykietą lub przez przejście po ostatnim rozkazie bloku do wykonania pierwszej instrukcji po nawiasie zamykającym.

Do bloku w kodzie maszyny można także wchodzić za pomocą instrukcji PROCEDURE n lub PROCEDURE I, jeżeli nawias otwierający, który rozpoczyna blok, jest poprzedzony etykietą, a wyjście z bloku prowadzi do instrukcji END.

10.2 Rozkazy

Rozkaz w kodzie maszyny składa się z trzech lub czterech części, mianowicie trzycyfrowej części operacyjnej OR (cyfry ósemkowe), pierwszego adresu AR, drugiego adresu NR i dwucyf-

rowej części modyfikacyjnej MR i ZR, którą pomija się w przypadku, gdy jest ona zerowa. Sąsiednie części rozkazu oddziela się pojedynczym odstępem. Adres (pierwszy lub drugi) może być liczbą dziesiętną całkowitą i nieujemną bez znaku (etykieta lub adres bezwzględny), taką samą liczbą zakończoną przecinkiem (adres względny liczony względem początku bloku w kodzie wewnętrznym; pierwsze słowo ma adres względny 0,; następne - adres 1, itd.), zmienną prostą lub zmienną ze wskaźnikiem liczbowym. Jeżeli drugi adres jest adresem względnym następnego słowa, to można go zastąpić znakiem +.

Przykład 10.2.1 Niżej podajemy kilka przykładowych rozkazów, wraz z krótkimi komentarzami.

050 3584 + (po wykonaniu - następny rozkaz bloku)

050 I 5 (po wykonaniu - instrukcja z etykietą 5)

237 X156 9, (po wykonaniu - rozkaz z komórki 9 licząc od początku bloku)

402 0 + 50 (po wykonaniu - następny rozkaz bloku)

000 0 10 (skutek taki, jak GO TO 10).

Operacje na zmiennych ze wskaźnikami zawierającymi zmienne można wykonywać korzystając z dowolnego rejestru modyfikacji i modyfikując część AR rozkazu.

Przykład 10.2.2 Przesłanie do akumulatora wartości zmiennej J(K+95) wykonuje się za pomocą rozkazów

```
050 K +
216 17 +
402 0 + 30
050 J95 + 31
```

Należy pamiętać, że translator przy tłumaczeniu bloku w kodzie wewnętrznym nie sprawdza formalnej poprawności napisanych rozkazów i np. nie będzie sygnalizował błędu przy tłumaczeniu rozkazu 237 I + , gdzie I jest zmienną typu całkowitego. Podobnie rozkaz 146 15 20 w programie, którego wstęp zawiera LABEL 10 nie będzie sygnalizowany jako błędny; skutki tego rozkazu można będzie przewidzieć tylko wtedy, gdy wiadomo z góry, jaka jest zawartość komórek o adresach 15 i 20 po wprowadzeniu do pamięci programu przetłumaczonego. Translator wykrywa natomiast takie błędy, jak ujemny adres, adres większy od 8191, cyfra 8 lub 9 w części operacyjnej rozkazu, zmienna nie opisana na początku programu itp.

10.3 Stałe

W bloku w kodzie wewnętrznym mogą również występować jako słowa maszynowe liczby napisane w sposób zgodny z uwagami z §2.3, jednak każdorazowo poprzedzone jedną z liter I,R,F określającą rodzaj liczby:

(a) I - liczba całkowita,

(b) R - liczba zmiennoprzecinkowa,

(c) F - ułamek stałoprzecinkowy, to jest co najwyżej 12-cyfrowy ułamek dziesiętny z przedziału $\langle -1, 1-2^{-38} \rangle$, który ma być zapamiętany w zwykłej postaci stałoprzecinkowej.

Przykład 10.3.1 Przykłady liczby w bloku w kodzie wewnętrznym:

I256
R256
R256.0
R-282.625'-8

F.31416
IO
FO
F-.2718
R2.26'

Należy pamiętać, że słowa maszynowe IO i RO są różne.

11. PISANIE I DZIURKOWANIE PROGRAMÓW W AUTOKODZIE

11.1 Postać programów w druku i na taśmie. Poprawianie omyłek.

Każdą instrukcję, nawiasy ograniczające bloki w kodzie wewnętrznym, każde słowo maszynowe w tych blokach, każdą z czterech części wstępu i zakończenie programu pisze się od początku nowego wiersza. Po każdym z wymienionych niżej słów i symboli daje się pojedynczy odstęp:

ABS ARCSIN ARCTAN BUTTON COS END (ale tylko w instrukcji kończącej pętlę) ENTIER EXP FOR FRAC GO IF INPUT INTEGER LABEL LN PRINT PRINTLINE PRINTOUT PRINTSPACE PROCEDURE PUNCH PUNCHLINE PUNCHOUT PUNCHSPACE READ REAL REPEAT SIN SQRT STAND START STEP STOP (ale tylko przed etykietą) TAN TEST TO UNTIL

Słowa UNTIL i REPEAT muszą być poprzedzone pojedynczym odstępem. Pojedynczy odstęp daje się ponadto po etykietach, jeżeli występują one wewnątrz instrukcji skoku. Poza tym odstępy na taśmach z programem można dawać jedynie w tekstach, które mają być drukowane (dziurkowane) przez program.

Taśma programu w autokodzie powinna zaczynać się od pewnej ilości znaków bl lub ls, co umożliwia właściwe założenie taśmy do czytnika maszyny; dalej musi być znak liter ls,

a przedtem lub potem mogą być znaki cr lf , po czym następują kolejne znaki składające się na kolejne instrukcje. Po każdej instrukcji dziurkuje się znaki cr lf. Daje to tę postać programu w druku, której wymaga się od tekstu przygotowanego ręcznie. Znaki fs i ls należy dziurkować wszędzie tam, gdzie to jest konieczne dla prawidłowej postaci programu w druku. Znaki cr i bl oraz zbędne znaki fs i ls są pomijane. To ostatnie można wykorzystać do poprawiania błędnie wydziurkowanych fragmentów taśmy przez zadziurkowywanie ich znakami liter. Jeżeli następne znaki są cyfrowe, to należy bezpośrednio potem wydziurkować znak cyfr fs. Jeżeli po ostatnim znaku lf nie była przeczytana żadna instrukcja, to wszystkie dalsze kolejne znaki lf są pomijane. To można wykorzystać do dowolnego dzielenia arkusza programu na stronice.

Jeżeli omyłka popełniona przy dziurkowaniu programu zostanie zauważona jeszcze przed wydziurkowaniem najbliższego znaku lf, to w celu anulowania tego wiersza dziurkuje się znak ?, po którym bezpośrednio można napisać poprawną wersję tej instrukcji zarówno po zmianie wiersza, jak i przed nią. Nie dotyczy to jednak znaków wydziurkowanych po znaku początku tekstu. Ten sposób poprawiania omyłek jest dopuszczalny także na taśmach z danymi. Tutaj znak ? anuluje wszystkie znaki od ostatniego symbolu końca liczby, tj. przecinka, podwójnego odstępu lub znaku lf.

11.2 Programy dziurkowane na wielu taśmach

Program może być wydziurkowany na wielu taśmach. Sygnałem końca taśmy, po której nastąpi inna taśma zawierająca dalszy

ciąg programu jest instrukcja NEXT. Po przeczytaniu tej instrukcji translator drukuje na dalekopisie znak oraz tę instrukcję, po czym zatrzymuje tłumaczenie. Po instrukcji NEXT może nastąpić komentarz dowolnej postaci, jednak nie zawierający więcej znaków, niż pozwala na to długość wiersza dalekopisu. Ten komentarz translator drukuje wraz ze słowem NEXT. Można to wykorzystać do dostarczania objaśnień operatorowi maszyny; komentarz może mieć np. postać zdania

ZALOZYC TASME PODPROGRAMU WYZNACZNIKA

Pamiętać należy, że instrukcja NEXT nie ma odpowiednika w programie przetłumaczonym, wobec czego opatrzenie jej etykietą jest zawsze pozbawione sensu.

Translator tłumaczy dalszy ciąg programu po napisaniu przez operatora na dalekopisie maszyny instrukcji TAPE zakończonej w zwykły sposób (cr lf lub lf). Instrukcja TAPE napisana na taśmie z programem jest pomijana przez translator.

12. DZIAŁANIE TRANSLATORA I PROGRAMÓW PRZETŁUMACZONYCH

12.1 Uwagi ogólne

Wszystkie istotne szczegóły dotyczące obsługi translatora i tworzonych przezeń programów podano w [2]. Informacje podane tutaj mają na celu zapoznać czytelnika z niektórymi ułatwieniami, które translator wprowadza do tłumaczenia programu i - później - do wykonywania obliczeń według tego programu.

12.2 Rodzaje pracy translatora

Translator może w czasie czytania programu w autokodzie wykonywać jedną z następujących czynności:

(a) dziurkowanie przekładu programu na taśmie (metoda pośrednia),

(b) zapamiętanie przekładu dla natychmiastowego wykonania (metoda bezpośrednia),

(c) analizę programu tylko dla sprawdzenia jego formalnej poprawności bez tworzenia przekładu w jakiejkolwiek postaci.

Informację o żądanym rodzaju pracy podaje się na pulpicie sterowania maszyny przed przystąpieniem do przetwarzania programu.

Każdą z wymienionych wyżej czynności translator może wykonywać krokowo, to znaczy zatrzymywać pracę po przeczytaniu i przetworzeniu każdego wiersza programu w autokodzie. Żądanie pracy krokowej podaje się ustawiając odpowiednią informację na pulpicie sterowania; tę informację można zmieniać w czasie pracy translatora. Następny krok pracy translator wykonuje po naciśnięciu przycisku START.

12.3 Informacje drukowane przez translator w czasie tłumaczenia programu poprawnego

W czasie czytania opisów zmiennych i tablic translator drukuje wyznaczone przez siebie adresy zmiennych prostych i początków tablic wyrażone w układzie ósemkowym. Ten układ został wybrany ze względu na łatwość zamiany liczb ósemkowych na dwójkowe.

Jeżeli tłumaczony program jest poprawny i jego przekład może zmieścić się w pamięci maszyny, to translator aż do zakończenia tłumaczenia nie drukuje żadnych informacji na dalekopisie z wyjątkiem ewentualnych instrukcji NEXT z komentarzami. Po zakończeniu tłumaczenia programu translator drukuje

adres ostatniego rozkazu i adres pierwszej stałej należącej do przekładu - obydwa wyrażone w układzie ósemkowym. Adresy komórek większe od pierwszego i mniejsze od drugiego z podanych są adresami wolnych komórek pamięci, tzn. nieużywanych przez program przetłumaczony.

Jeżeli przekład programu mógłby nie zmieścić się w pamięci, to translator drukuje znaki ?? i zatrzymuje maszynę. Ponowne jej uruchomienie powoduje kontynuację tłumaczenia - być może, w błędny sposób. Tak wymuszone tłumaczenie może nie mieć żadnej wartości praktycznej poza tym tylko, że na podstawie wydrukowanych przez translator adresów można obliczyć brakującą liczbę komórek pamięci.

12.4 Sygnalizacja omyłek w programach

Po wykryciu w czytanym wierszu błędu typu składniowego (na przykład instrukcji I=SIN J albo GO TO 2 4 IF BUTTON 31 i podobnych) translator drukuje znak 卐 , a następnie ten wiersz i przestawia się na czytanie dalszego ciągu programu z dalekopisu. Jeżeli błąd da się naprawić bez zmiany poprzednich wierszy uznanych przez translator za poprawne, to należy wprowadzić z klawiatury dalekopisu odpowiednie poprawki (poprawną wersję tego wiersza lub większy fragment programu), a po ostatnim wierszu napisać instrukcję TAPE. Każdy wiersz poprawiony i instrukcję TAPE można wprowadzać dopiero po wydrukowaniu przez translator znaku 卐 . Instrukcja TAPE przestawia translator na czytanie z czytnika, trzeba więc przed jej wydrukowaniem odpowiednio ustawić taśmę. Jeżeli pierwsza instrukcja, która będzie czytana z taśmy, zaczyna się od znaku cyfrowego,

a nie ma przed nim znaku cyfr, to znak ten należy wydrukować bezpośrednio po słowie TAPE, przed zmianą wiersza. Jeżeli błąd wykryto w opisie zmiennych i tablic, tzn. w jakimkolwiek wierszu zaczynającym się od słowa REAL lub słowa INTEGER, to tłumaczenie programu należy powtórzyć od początku po poprawieniu początku taśmy.

Możliwe jest, że translator wydrukuje na dalekopisie instrukcję, w której nie ma żadnego błędu. Jeżeli nie jest to instrukcja NEXT, to taka sygnalizacja oznacza, że na miejscu wydrukowanej instrukcji powinna być jakaś inna instrukcja. Jeżeli, na przykład, w programie zostanie opuszczone słowo BEGIN, to każda instrukcja, która może wystąpić tylko poniżej tego słowa będzie sygnalizowana jako błędna. Podobnie instrukcja START n będzie traktowana jako błędna, jeżeli w programie nie zakończono wszystkich pętli (brak jakiejś instrukcji zaczynającej się od END). Takich przykładów można znaleźć znacznie więcej.

Translator sygnalizuje wszystkie możliwe błędy we fragmentach instrukcji złożonych ze zmiennych, znaków działań i liczb, np. niedopuszczalne mieszanie typów zmiennych w wyrażeniach i instrukcjach arytmetycznych, używanie wskaźników niezerowych lub zmiennych przy zmiennych prostych, użycie nie opisanej zmiennej lub etykiety itp. Natomiast translator nie wykrywa niektórych błędów w słowach używanych do pisania programów. Przy rozpoznawaniu tych słów translator zakłada, że są napisane poprawnie i identyfikuje je badając najmniejszą potrzebną do tego celu ilość znaków. Tak więc kontrola "ortografii" programu wykonywana przez translator ma charakter i skuteczność kontroli statystycznej i nie jest zupełnie wykluczone, chociaż

bardzo mało prawdopodobne, że pewna zupełnie bezsensowna kombinacja znaków będzie uznana przez translator za jakąś instrukcję w autokodzie i przetłumaczona. Jako przykład i pewnego rodzaju ostrzeżenie podajemy, że ciąg "instrukcji"

```
ORF X=Y SKOK Z URWAC U
I=I+1
XI=-APPSIK X
DEN X
```

będzie potraktowany przez translator dokładnie tak samo, jak ciąg poprawnych instrukcji

```
FOR X=Y STEP Z UNTIL U
I=I+1
XI=-ARCSIN X
END X
```

12.5 Sygnalizacja omyłek w programach utworzonych przez translator

Niektóre błędy programowania są wykrywane i sygnalizowane dopiero w czasie wykonywania programu. Oto przykłady: ujemna wartość zmiennej będącej argumentem logarytmu lub pierwiastka, instrukcja nadania zmiennej całkowitej wartości napisanej na taśmie z danymi z użyciem kropki dziesiętnej, przekroczenie dopuszczalnej liczby jednocześnie wykonywanych podprogramów itp. Sygnalizacja tych błędów polega zwykle na zatrzymaniu maszyny i wyświetleniu na pulpicie sterowania numeru przyporządkowanego wykrytemu błędowi. Wykaz tych błędów wraz z przyporządkowanymi numerami znajduje się w [2]. Ponowne uruchomienie maszyny zatrzymanej wskutek wykrycia błędu tego typu powoduje ponowne zatrzymanie z wyświetleniem tego samego numeru.

Niektóre błędy, np. użycie instrukcji

```
I=0
GO TO I
```

(§7.2) powodują zatrzymanie się maszyny, ale po ponownym uruchomieniu wykonuje ona program od początku. Inne dosyć oczywiste błędy, np. instrukcje

```
I=-5
GO TO I
```

albo instrukcje

```
I=-1
PROCEDURE I
```

powodują dezorganizację programu.

12.6 Ręczne sterowanie programami przetłumaczonymi

Układając programy w autokodzie warto pamiętać o możliwościach ręcznego sterowania tymi programami po ich przetłumaczeniu i umieszczeniu w pamięci maszyny. Obliczenia można rozpocząć w zwykły sposób, od instrukcji opatrzonej etykietą napisaną po słowie START. Można także rozpocząć program od dowolnej innej instrukcji opatrzonej etykietą przy założeniu, że etykieta ta nie jest napisana przed instrukcją stanowiącą część jakiegoś podprogramu wywoływanego instrukcją PROCEDURE (por. §7.7). Wykonywanie programu można ponadto przerwać w sposób opisany w §7.1 (po przykładzie 7.1.1, p.1°). Umiejętne stosowanie tych możliwości może przynieść znaczne ułatwienia przy korzystaniu z programów bardziej rozbudowanych, przy kontroli programów itp.

13. PRZYKŁADY PROGRAMÓW I PODPROGRAMÓW

W tym rozdziale podajemy kilka przykładów i podprogramów napisanych w autokodzie MOST 1, jednak bez wyjaśnień wielu szczegółów samego programowania. Uważne prześledzenie działania poszczególnych programów dostarczy czytelnikowi wielu informacji o praktycznych szczegółach opisywania algorytmów w języku MOST 1. Opis każdego przykładu zawiera tylko sformułowanie zadania i pewne informacje o zastosowanej metodzie numerycznej, a w przypadkach koniecznych - niezbędne informacje o samym programie.

18.1 Rozwiązywanie równania Keplera

Równanie $x - e \cdot \sin x = m$ nazywa się równaniem Keplera. Program oblicza i drukuje w postaci tablicy wartości niewiadomej x dla parametrów e i m zmieniających się od 0.1 co 0.1 do 0.6, po czym sprawdza obliczone rozwiązania i drukuje w postaci podobnej tablicy wartości lewej strony równania w wyznaczonych uprzednio punktach. W programie zastosowano metodę iteracyjną Newtona. Jako przybliżenie początkowe program przyjmuje w każdym przypadku liczbę $\frac{1}{2}(m + \pi)$. Proste rozważanie geometryczne wykazuje, że dla przyjętych w przykładzie (i pewnych innych) wartości parametrów e i m metoda Newtona jest dla tego przybliżenia początkowego zawsze zbieżna. Program wykonuje obliczenia z dokładnością zapewniającą poprawność wszystkich wydrukowanych cyfr.

Program 13.1

```
INTEGER I
REAL EMX36Y1Q
LABEL 3
```

```
BEGIN
1:PRINTLINE 5
PRINTSPACE 20
PRINT ⍾ROZWIAZANIA ROWNANIA KEPLERA⍾
PROCEDURE 3
I=0
Q=.00005
FOR M=.1 STEP .1 UNTIL .6
PRINTLINE 2
PRINT M,0.1
PRINTSPACE 3
FOR E=.1 STEP .1 UNTIL .6
I=I+1
X=M+3.1416
X=X×.5
2:Y1=COS X
Y1=Y1×E
Y1=Y1-1
Y=SIN X
Y=Y×E
Y=Y-X
Y=Y+M
Y=Y/Y1
X=X-Y
GO TO 2 0 0 IF Q=ABS Y
PRINT X,1.4
XI=X
END E
END M
PRINTLINE 5
PRINTSPACE 25
PRINT ⍾SPRAWDZENIE⍾
PROCEDURE 3
I=0
FOR M=.1 STEP .1 UNTIL .6
PRINTLINE 2
PRINT M,0.1
PRINTSPACE 3
FOR E=.1 STEP .1 UNTIL .6
```

```
I=I+1
Y=SIN XI
Y=Y×E
Y=XI-Y
PRINT Y,1.4
END E
END M
STOP
3:PRINT ft cr 1f^3 cr sp^4 E sp^4.1 sp^7 .2 sp^7 .3 sp^7 .4 sp^7
                .5 sp^7 .6 cr 1f cr sp^2 M cr 1f^2 cr ft
END
START 1
```

Wyniki wydrukowane przez program 13.1

ROZWIAZANIA RÓWNANIA KEPLERA

E / M	.1	.2	.3	.4	.5	.6
.1	0.1111	0.1249	0.1427	0.1662	0.1987	0.2463
.2	0.2220	0.2494	0.2841	0.3294	0.3902	0.4737
.3	0.3327	0.3729	0.4232	0.4873	0.5697	0.6749
.4	0.4429	0.4950	0.5591	0.6383	0.7355	0.8513
.5	0.5525	0.6155	0.6913	0.7818	0.8879	1.0072
.6	0.6614	0.7340	0.8192	0.9177	1.0282	1.1469

SPRAWDZENIE

E / M	.1	.2	.3	.4	.5	.6
.1	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
.2	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000
.3	0.3000	0.3000	0.3000	0.3000	0.3000	0.3000
.4	0.4000	0.4000	0.4000	0.4000	0.4000	0.4000
.5	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
.6	0.6000	0.6000	0.6000	0.6000	0.6000	0.6000

Przy reprodukowaniu wyników dostarczonych przez maszynę nie zachowano rzeczywistych odstępów między wierszami, które daje dalekopis maszyny. Przy projektowaniu formatu arkusza z wynikami trzeba pamiętać, że odstęp między wierszami zależ od aktualnego ustawienia dalekopisu i nie można tego odstępu łatwo zmienić.

13.2 Podprogram obliczania wyznacznika macierzy

Przytoczony dalej podprogram można włączyć do każdego programu napisanego w autokodzie pod warunkiem, że nie są w nim używane etykiety 1,2,3,4 i 5, a opis zmiennych zawiera także zmienne całkowite JLMN8 i tablice zmiennoprzecinkowe B2, Am, $m=k^2$, gdzie k jest stopniem macierzy. Przed wejściem do podprogramu należy nadać zmiennej N1 wartość k, a kolejne elementy macierzy należy umieścić w tablicy A, od A1 do Am. Po wyjściu z podprogramu aktualna wartość zmiennej A jest równa wartości wyznacznika; aktualne wartości innych zmiennych z tablicy A zostaną w czasie obliczeń zmienione podobnie, jak wartość zmiennej N1.

Podprogram działa według dobrze znanej metody sprowadzania macierzy do postaci trójkątnej (górnej), przy czym w celu zapewnienia odpowiedniej dokładności obliczeń wybiera się w każdym etapie obliczeń największy element w kolejnej kolumnie macierzy, po czym przestawia się wiersze macierzy w ten sposób, aby ten element wprowadzić na przekątną. Dla wyzerowania elementów podprzekątniowych w danej kolumnie trzeba kolejny wiersz mnożyć przez odpowiednie współczynniki i odejmować odpowiednio od wszystkich wierszy leżących poniżej. Przy opisanym przestawianiu wierszy współczynniki te

mają moduły mniejsze od jedności, co - jak można pokazać - zapewnia największą możliwą dokładność obliczeń. Wartość wyznacznika ma zwykle nie mniej niż 7 cyfr dokładnych.

Wywołania podprogramu dokonuje się przy pomocy instrukcji PROCEDURE 1.

Podprogram 18.2

```
1:N=N1×N1
N3=N1+1
N4=N1-1
N2=N1
A=1
FOR L=1 STEP N3 REPEAT N4
N5=L+1
N6=N2-1
B=ABS AL
N7=L
B1=AL
FOR J=N5 STEP 1 REPEAT N6
GO TO 0 2 2 IF B=ABS AJ
B=ABS AJ
B1=AJ
N7=J
2:END J
A=A×B1
GO TO 0 5 0
GO TO 0 3 0 IF N7=L
A=-A
N8=N7-L
FOR J=L STEP N1 REPEAT N2
M=N8+J
B2=AJ
AJ=AM
AM=B2
END J
3:N7=L+N1
```

```
FOR M=1 STEP 1 REPEAT N6
B1=A(L+M)
GO TO 0 4 0
B2=B1/AL
FOR J=N7 STEP N1 REPEAT N6
B=AJ×B2
A(J+M)=A(J+M)-B
END J
4:END M
N2=N2-1
END L
A=A×AN
5:END
NEXT KONIEC PODPROGRAMU WYZNACZNIKA
```

Przytoczony podprogram jest przekładem z autokodu MARK III podprogramu (w ulepszonej wersji) M11 z biblioteki podprogramów maszyny cyfrowej ELLIOTT 803.

13.3 Rozwiązanie równania trzeciego stopnia metodą Warmusa

Niżej podajemy kompletny program rozwiązywania równania trzeciego stopnia $ax^3 + bx^2 + cx + d = 0$ metodą Warmusa, opisaną w t. VI Zastosowań Matematyki.

Zasadniczą procedurę numeryczną oddzielono kilkoma pustymi wierszami od reszty programu.

Program 13.3

```
REAL A1B1CDQUVWST
LABEL 12
BEGIN
1:READ A
READ B
READ C
READ D
```

```
A1=3×A
B1=B/A1
A1=C/A1
W=B1×B1
W=W-A1
A1=A1×B1
S=D/A
A1=A1-S
V=B1+B1
V=V×W
V=V-A1
GO TO 2 0 2
T=-B1
GO TO 8
2:U=ABS V
U=LN U
U=U/3
U=EXP U
Q=U×U
GO TO 0 3 3 IF V=0
U=-U
3:GO TO 0 4 4 IF W=0
T=1.71429×W
T=T-Q
T=V/T
T=T-B1
GO TO 7
4:GO TO 5 0 5 IF U=0
T=0
GO TO 6
5:T=3×W
T=T+Q
T=SQRT T
GO TO 0 6 6 IF U=0
T=-T
6:S=.05822×U
S=S-B1
```

```
T=S-T
7:FOR V=0,0,0,0
U=B1+T
U=U×U
T=T+T
T=T×U
T=T+A1
S=U-W
T=T/S
T=T-B1
T=T/3
END V
8:A1=T+B1
A1=A1×.5
Q=-A1-B1
A1=A1×A1
A1=W-A1
A1=A1×3
V=ABS A1
V=SQRT V
GO TO 9 0 0 IF A1=0
U=Q+V
V=Q-V
GO TO 10
9:U=Q

10:PRINTLINE 6
PRINTSPACE 33
PRINT ℜ3 sp^4 2 cr lf cr sp^10 ROZWIĄZANIA sp RO cr^2 sp^15 fs ,
       sp^8 ' ls WNANIA sp AX sp + sp BX sp + sp CX sp +
       sp D sp = sp 0 sp , sp^2 GDZIE cr lf^2 cr A sp = sp^14
       B sp = sp^14 C sp = sp^14 D sp = cr^2ℜ
PRINTSPACE 3
PRINT A,6'
PRINTSPACE 3
PRINT B
```

```
PRINTSPACE 3
PRINT C
PRINTSPACE 3
PRINT D
PRINTLINE 3
PRINT ₰PIERWIASTKI: sp⁹ X1₰
GO TO 11 0 0 IF A1=0
PRINT ₰sp¹³ X2 sp¹³ X3₰
GO TO 12
11:PRINT ₰sp⁷ RE(X2 sp I sp X3) sp³ IM(X2 sp I X3)₰
12:PRINTLINE 2
PRINTSPACE 15
PRINT T
PRINT U
PRINT V
PRINTLINE 6
STOP
START 1
```

Przykładowe wyniki wydrukowane przez program 13.3

ROZWIĄZANIA RÓWNANIA $AX^3 + BX^2 + CX + D = 0$, GDZIE

A = .100000' 01 B = .200000' 01 C = .200000' 01 D = .100000' 01

PIERWIASTKI:	X1	RE(X2 I X3)	IM(X2 I X3)
	.100000' 01	-.500000' 00	.866025' 00

Prace cytowane

[1] Maszyna cyfrowa ODRA 1003. Opis ogólny

[2] Instrukcja obsługi translatora MOST 1.