WOJSKOWA AKADEMIA TECHNICZNA
im. Jarosława Dąbrowskiego
INSTYTUT AUTOMATYZACJI SYSTEMÓW ZARZĄDZANIA

ALGOL 60

dla

ZAM–41

WARSZAWA 1969

O P R O G R A M O W A N I E
UNIWERSALNEJ MASZYNY CYFROWEJ ZAM-41

S Y S T E M O P E R Y C Y J N Y 1 4 1
A L G O L 6 0
/do użytku wewnętrznego/

OPRACOWAŁ ZESPÓŁ
pod kierownictwem mgr M.Łąckiej
w składzie:

mgr inż. S. Choromański
mgr J. Koncewicz
mgr E. Kozłowska
mgr inż. W. Romaniuk
mgr inż. Z. Zorski

INSTYTUT MASZYN MATEMATYCZNYCH
WARSZAWA 1969

1. WSTĘP.

Język ALGOL, który poniżej opisujemy jest konkretną reprezentacją języka ALGOL-60 zdefiniowaną dla maszyny ZAM-41. Definicje jednostek składniowych opisywanej reprezentacji języka ALGOL oraz reguły tworzenia w tym języku poprawnych struktur podajemy w rozdziałach 2-5, które są tłumaczeniem odpowiednich rozdziałów oficjalnego dokumentu: "Revised Report on the Algorithmic Language ALGOL 60"[1/], uwzględniającym zmiany wynikające z przyjętej reprezentacji. Niniejszy opis opiera się w zasadzie na tłumaczeniu powyższego dokumentu przez S.Paszkowskiego[2/] z pewnymi modyfikacjami terminologii zatwierdzonymi przez zespół C Komisji d/s Systemów Oprogramowania Maszyn Cyfrowych[3/]. Pozostałe zmiany terminologii, które nastąpiły w stosunku do wymienionych prac wynikają z konieczności zastosowania jednolitej terminologii w opisach wszystkich elementów oprogramowania maszyny ZAM-41. Standardowe procedury wejścia-wyjścia ALGOLu opisane są w rozdziale 6. Rozdział 7 zawiera informacje dotyczące uruchamiania programów napisanych w ALGOLu w Systemie Oprogramowania 141.

1/ Naur, P. /ed./, Revised Report on the Algorithmic Language ALGOL 60, IFIP, Copenhagen 1962.

2/ Paszkowski, St., Język ALGOL 60, PWN, Warszawa 1968, Dodatek A str 213-253.

3/ Protokół z posiedzenia zespołu C Komisji d/s Systemów Oprogramowania Maszyn Cyfrowych odbytego dnia 8 maja 1968 w Zakopanem w czasie Konferencji GAMS GAJAPEI.

Translator języka ALGOL pracujący na maszynie ZAM-41 jest adaptacją translatora Whetstone Compiler, opracowanego dla maszyny KDF9[4/], zrealizowaną z wieloma modyfikacjami wynikającymi ze specyfiki maszyny ZAM-41. Tłumaczenie programu źródłowego na program wynikowy odbywa się w dwu etapach, zwanych odpowiednio przebiegiem pierwszym i drugim. Program wynikowy otrzymuje się w specjalnym języku makrorozkazów wykonywanych przez program zwany Interpretatorem. Rozmieszczenie danych w pamięci operacyjnej dokonuje się dynamicznie /podczas wykonywania programu wynikowego/ za pomocą programowanego stosu. Zarówno program wynikowy, jak i stos są automatycznie segmentowane i umieszczane w pamięci bębnowej, skąd Interpretator sprowadza je do pamięci operacyjnej.

Składnia języka /tj. obiekty określone w języku/ została opisana w notacji zaproponowanej przez J.W.Backusa[5/], - za pomocą formuł metajęzykowych. Formuły metajęzykowe składają się ze zmiennych metajęzykowych, stałych metajęzykowych i łączników metajęzykowych.

Są dwa łączniki metajęzykowe; oznaczone są one znakami ::= oraz | , a czytane odpowiednio jako "równe z definicji" i "lub".

4/ Randell, B. i Russell, L.J., ALGOL 60 Implementation, AP, London and New York 1964.

5/ Backus, J.W., The syntax and semantics of the proposed international algebraic language of the Zurich - ACM GAMM conference, ICIP. Paris, June 1959.

2.3. Ograniczniki.

⟨ogranicznik⟩ ::= ⟨operator⟩|⟨separator⟩|⟨nawias⟩|
⟨deklarator⟩|⟨specyfikator⟩

⟨operator⟩ ::= ⟨operator arytmetyczny⟩|⟨operator relacji⟩|
⟨operator logiczny⟩|⟨operator następstwa⟩

⟨operator arytmetyczny⟩ ::= +|-|*|/| DIV| POWER

⟨operator relacji⟩ ::= LESS|NOTGREATER|=|EQUAL|NOTLESS|
GREATER|NOTEQUAL

⟨operator logiczny⟩ ::= EQUIV|IMPL|OR|AND|NOT

⟨operator następstwa⟩ ::= GOTO|GO TO|IF|THEN|ELSE|FOR|DO

⟨separator⟩ ::= ,|.|E|:|;|:=|STEP|UNTIL|WHILE|COMMENT
⟨spacja⟩|⟨nowa linia⟩

⟨nawias⟩ ::= (|)|[|]|'|BEGIN|END

⟨deklarator⟩ ::= OWN|BOOLEAN|INTEGER|REAL|ARRY|SWITCH|
PROCEDURE

⟨specyfikator⟩ ::= STRING|LABEL|VALUE

Ograniczniki mają ustalone znaczenie, które w większości przypadków jest oczywiste, a w pozostałych przypadkach będzie wyjaśnione w odpowiednim miejscu /patrz również Dodatek A/.

2.3.1. Słowa zastrzeżone.

Ograniczniki będące słowami utworzonymi z liter oraz wartości logiczne są traktowane jako słowa zastrzeżone. Oznacza to, że nie można ich używać jako identyfikatorów.

2. SYMBOLE PODSTAWOWE, IDENTYFIKATORY, LICZBY, TEKSTY I SŁOWA ZASTRZEŻONE. POJĘCIA PODSTAWOWE.

Język ALGOL jest zbudowany z następujących symboli podstawowych:

⟨symbol podstawowy⟩ ::= ⟨litera⟩|⟨cyfra⟩|⟨wartość logiczna⟩|⟨ogranicznik⟩

2.1. Litery.

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

Litery nie mają indywidualnej wartości. Używa się ich do tworzenia identyfikatorów, tekstów i słów zastrzeżonych /patrz punkty 2.3. "Ograniczniki", 2.4. "Identyfikatory" oraz 2.6. "Teksty"/.

2.2.1. Cyfry.

⟨cyfra⟩ ::= 0|1|2|3|4|5|6|7|8|9

Cyfry służą do tworzenia liczb, identyfikatorów i tekstów.

2.2.2. Wartości logiczne.

⟨wartość logiczna⟩ ::= TRUE|FALSE

Sens wartości logicznych jest oczywisty /patrz Dodatek A/.

Znak ::= dzieli formułę na dwie części. Z jego lewej strony umieszczany jest obiekt definiowany, a z prawej strony obiekty definiujące. W wyniku każdy z obiektów jest zdefiniowany przez tak zwane symbole podstawowe języka (litery, cyfry, nawiasy, znaki interpunkcji itp). Symbole podstawowe w formułach metajęzykowych reprezentują siebie same, podczas gdy inne oznaczenia reprezentują zawsze całą klasę obiektów. Z tego względu symbole podstawowe zwane są stałymi metajęzykowymi, a pozostałe obiekty - zmiennymi metajęzykowymi. Zmienne metajęzykowe w formułach metajęzykowych zamykane są w nawiasy kątowe. Symboli podstawowych nie zamyka się w nawiasy kątowe.

Na przykład, formuła

<ab> ::= (|[|<ab>(|<ab><d>

podaje rekurencyjną regułę tworzenia wartości zmiennej <ab>. Wynika z niej, że zmienna <ab> może mieć wartość

(

lub

[

lub

jeżeli dana jest pewna dopuszczalna wartość <ab> , to nową jej wartość można otrzymać dopisując do niej znak (

lub

jeżeli dana jest pewna dopuszczalna wartość <ab> , to nową jej wartość można otrzymać dopisując do niej pewną wartość zmiennej <d> .

Niech warościami <d> są cyfry dziesiętne

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

wówczas niektórymi wartościami zmiennej metajęzykowej ab są:

[(
(((
[(((1
[(((1(
[(((1(37
(12345
[86

Definicja:

<puste> ::=

Jest to pusty ciąg znaków.

przełączników i procedur. Można je wybierać dowolnie, jednak z uwzględnieniem ograniczeń wyliczonych w punktach 2.3.1. "Słowa zastrzeżone" oraz 3.2.4. "Funkcje standardowe". Wewnątrz identyfikatora nie może występować ani spacja ani nowa linia.

Jeżeli po identyfikatorze nie następuje żaden z ograniczników różnych od słów zastrzeżonych, to należy po nim umieścić spację lub nową linię.

Różne wielkości można oznaczać tym samym identyfikatorem tylko wtedy, gdy mają rozłączne obszary działania, określone w programie deklaracjami /patrz punkt 2.7. "Wielkości, klasy i obszary działania" oraz rozdział 5. "Deklaracje"/.

2.4.4. Ograniczenie.

Liczba różnych identyfikatorów deklarowanych w programie została ograniczona do 511. Maksymalna długość identyfikatora nie może przekraczać 68 liter i/lub cyfr.

2.5. Liczby.

2.5.1. Składnia.

⟨liczba całkowita bez znaku⟩ ::= ⟨cyfra⟩|
⟨liczba całkowita bez znaku⟩
⟨cyfra⟩

⟨liczba całkowita⟩ ::= ⟨liczba całkowita bez znaku⟩|
+⟨liczba całkowita bez znaku⟩|

- ⟨liczba całkowita bez znaku⟩

⟨ułamek dziesiętny⟩ ::= . ⟨liczba całkowita bez znaku⟩

⟨cecha⟩ ::= E⟨liczba całkowita⟩

⟨liczba dziesiętna⟩ ::= ⟨liczba całkowita bez znaku⟩|
⟨ułamek dziesiętny⟩|
⟨liczba całkowita bez znaku⟩
⟨ułamek dziesiętny⟩

⟨liczba bez znaku⟩ ::= ⟨liczba dziesiętna⟩|
⟨liczba dziesiętna⟩⟨cecha⟩

⟨liczba⟩ ::= ⟨liczba bez znaku⟩| + ⟨liczba bez znaku⟩|
- ⟨liczba bez znaku⟩

2.5.2. Przykłady.

0	-200.084	-.083E-02
1777	+07.43	.5384
+0.7300	9.34E+10	2E-4

2.5.3. Znaczenie.

Liczby dziesiętne mają swe zwykłe znaczenie. Cecha jest czynnikiem skalującym, równym potędze liczby 10 o wykładniku całkowitym. Wewnątrz liczby nie może występować ani spacja, ani nowa linia. Jeżeli po liczbie nie następuje żaden ogranicznik spośród ograniczników różnych od słów zastrzeżonych, to należy po niej umieścić spację lub nową linię.

Wewnątrz słowa zastrzeżonego nie może występować spacja ani nowa linia. Między słowami GO i TO może pojawić się dokładnie jedna spacja, ale nie może występować nowa linia. Jeżeli po słowie zastrzeżonym nie następuje żaden ogranicznik spośród ograniczników różnych od słów zastrzeżonych, to należy po nim umieścić spację lub nową linię. Spis słów zastrzeżonych i ich znaczenia w języku polskim podany jest w Dodatku A.

Przyjmuje się, że następujące formy pisania ograniczników są równoważne:

GOTO GO TO

EQUAL =

Między symbolami ":" i "=" w ograniczniku ":=" nie może występować spacja, ani nowa linia.

UWAGA: Litery, cyfry oraz ograniczniki różne od słów zastrzeżonych będziemy dalej nazywali często symbolami znakowymi albo po prostu znakami.

2.3.2. Komentarze.

Następujące reguły pozwalają wstawić komentarz między symbolami programu:

Ciąg symboli podstawowych:	Jego równoważnik
; COMMENT ⟨ciąg dowolnych znaków nie zawierający ;⟩ ;	;
BEGIN COMMENT ⟨ciąg dowolnych znaków	

nie zawierający ;> ; BEGIN

END <ciąg dowolnych znaków nie zawierający END ani ; ani ELSE> END

Równoważność oznacza tu, że każdą z trzech struktur podanych w lewej kolumnie, występującą poza tekstem, można zastąpić odpowiednim symbolem z prawej kolumny bez jakiegokolwiek wpływu na program. Przyjmuje się też, że pierwszy /przy czytaniu tekstu programu od lewej do prawej/ komentarz ma pierwszeństwo w zastępowaniu go równoważnikiem przed strukturami znajdującymi się w dalszym ciągu programu.

2.4. Identyfikatory.

2.4.1. Składnia.

<identyfikator> ::= <litera>|<identyfikator><litera>|
<identyfikator><cyfra>

2.4.2. Przykłady.

A

BETA

ALFA5

PIERWIASTKIWIELOMIANU

D12MP32F

2.4.3. Znaczenie.

Identyfikatory nie mają samoistnego sensu, lecz służą jedynie do nazywania zmiennych prostych, macierzy, etykiet,

2.5.4. Typy.

Liczby całkowite są typu INTEGER. Wszystkie inne liczby są typu REAL /patrz punkt 5.1. "Deklaracje zmiennych prostych"/.

2.6. Teksty.

2.6.1. Składnia.

⟨tekst⟩ ::= '⟨dowolny ciąg symboli znakowych nie zawierający '⟩'

2.6.2. Przykłady.

'I'

'5K,-] =+([/17T)125'

'PIERWIASTKI: 2BWIELOMIANU'

2.6.3. Znaczenie.

Teksty wykorzystywane są przez standardowe procedury wyjścia /punkt 6.1. "Instrukcje wyjścia"/.

Tekst może zawierać co najwyżej 384 symboli znakowych, wliczając w to obydwa symbole " '".

Wewnątrz tekstu symbole spacji i nowej linii nie mają znaczenia.

2.7. Wielkości, klasy i obszary działania

Rozróżnia się następujące klasy wielkości: zmienne proste, macierze, etykiety, przełączniki i procedury.
Obszarem działania wielkości jest zbiór instrukcji i wyrażeń,

w którym obowiązuje deklaracja identyfikatora tej wielkości. Odnośnie etykiet, patrz punkt 4.1.3.

2.8. Wartości i typy.

Wartością nazywamy uporządkowany zbiór liczb /w szczególności jedną liczbę/, uporządkowany zbiór wartości logicznych /w szczególności jedną wartość logiczną/ lub etykietę.

Niektóre jednostki składniowe mogą przybierać wartości, które na ogół zmieniają się w czasie wykonywania programu. Wartości wyrażeń i ich składowych zostaną zdefiniowane w rozdziale 3.

Wartością identyfikatora macierzy jest uporządkowany zbiór wartości odpowiedniej macierzy zmiennych indeksowanych /punkt 3.1.4.1/.

Poszczególne typy INTEGER, REAL, BOOLEAN oznaczają ogólne cechy wartości. Typy związane z jednostkami składniowymi odnoszą się do wartości tych jednostek.

3. WYRAŻENIA.

Podstawowymi częściami składowymi programów napisanych w języku ALGOL są wyrażenia arytmetyczne, boolowskie i sterujące. Elementami tych wyrażeń, oprócz ograniczników, są wartości logiczne, liczby, zmienne, odwołania funkcyjne oraz operatory elementarne: arytmetyczne, relacji, logiczne i następstwa. Ponieważ w definicjach syntaktycznych zmiennej i odwołania funkcyjnego występuje wyrażenie, to definicja wyrażenia i jego elementów jest z konieczności rekurencyjna.

⟨wyrażenie⟩ ::= ⟨wyrażenie arytmetyczne⟩|
⟨wyrażenie boolowskie⟩|
⟨wyrażenie sterujące⟩

3.1. Zmienne.

3.1.1. Składnia.

⟨identyfikator zmiennej⟩ ::= ⟨identyfikator⟩

⟨zmienna prosta⟩ ::= ⟨identyfikator zmiennej⟩

⟨wyrażenie indeksowe⟩ ::= ⟨wyrażenie arytmetyczne⟩

⟨lista indeksów⟩ ::= ⟨wyrażenie indeksowe⟩|⟨lista indeksów⟩,
⟨wyrażenie indeksowe⟩

⟨identyfikator macierzy⟩ ::= ⟨identyfikator⟩

⟨zmienna indeksowana⟩ ::= ⟨identyfikator macierzy⟩
[⟨lista indeksów⟩]

⟨zmienna⟩ ::= ⟨zmienna prosta⟩|⟨zmienna indeksowana⟩

3.1.2. Przykłady.

```
DELTA
A17
Q[7,2]
X[SIN(N*PI/2), Q[3, N, 4], A]
```

3.1.3. Znaczenie.

Zmienna jest nazwą, nadaną pojedyńczej wartości. Wartość tę można wykorzystać w wyrażeniach do tworzenia innych wartości i można ją zmienić za pomocą instrukcji przypisania /patrz punkt 4.2./. Typ wartości zmiennej określany jest deklaracją tej zmiennej /patrz punkt 5.1. "Deklaracje zmiennych prostych"/ lub odpowiednią deklaracją identyfikatora macierzy /patrz punkt 5.2. "Deklaracje macierzy"/.

3.1.4. Indeksy.

3.1.4.1. Zmienne indeksowane są nazwami wartości elementów macierzy /patrz punkt 5.2. "Deklaracje macierzy"/. Każde wyrażenie indeksowe, znajdujące się na liście indeksów, zajmuje jedną pozycję indeksu w zmiennej indeksowanej i nazywa się indeksem. Pełną listę indeksów ujmuje się w nawiasy kwadratowe "[" "]". Element macierzy, odpowiadający zmiennej indeksowanej, jest określany przez aktualne wartości liczbowe indeksów /patrz punkt 3.3. "Wyrażenia arytmetyczne"/.

3.1.4.2. Każdy indeks interpretuje się jako zmienną typu INTEGER, a przez obliczenie indeksu rozumie się przypisanie wartości tej fikcyjnej zmiennej /punkt 4.2.4./.
Wartość zmiennej indeksowanej jest określona tylko wtedy, gdy wartość wyrażenia indeksowego zawiera się w granicach indeksu macierzy /patrz 5.2. "Deklaracje macierzy"/.

3.1.4.3. Bezwzględna wartość indeksu nie może przekraczać liczby 32767.

3.2. Odwołanie funkcyjne.

3.2.1. Składnia.

⟨identyfikator procedury⟩ ::= ⟨identyfikator⟩

⟨parametr aktualny⟩ ::= ⟨tekst⟩|⟨wyrażenie⟩|
⟨identyfikator macierzy⟩|
⟨identyfikator przełącznika⟩|
⟨identyfikator procedury⟩

⟨tekst literowy⟩ ::= ⟨litera⟩|⟨tekst literowy⟩ ⟨litera⟩

⟨ogranicznik parametru⟩ ::= ,|)⟨ tekst literowy⟩ :(

⟨lista parametrów aktualnych⟩ ::= ⟨parametr aktualny⟩|
⟨lista parametrów aktualnych⟩
⟨ogranicznik parametru⟩
⟨parametr aktualny⟩

⟨zbiór parametrów aktualnych⟩ ::= ⟨puste⟩|(⟨lista parametrów aktualnych⟩)

⟨odwołanie funkcyjne⟩ ::= ⟨identyfikator procedury⟩ ⟨zbiór parametrów aktualnych⟩

3.2.2. Przykłady.

SIN(A+B)

F(A,B-C)

S(X-Y) TEMPERATURA:(T)CIŚNIENIE:(C)

3.2.3. Znaczenie.

Odwołanie funkcyjne określa pojedyńczą wartość liczbową lub logiczną, która jest wynikiem zastosowania odpowiedniego ciągu reguł zadanych deklaracją procedury /punkt 5.4. "Deklaracje procedur"/ do ustalonego zbioru parametrów aktualnych. Reguły rządzące określaniem parametrów aktualnych podane są w punkcie 4.7. /"Instrukcje procedury"/. Nie każda deklaracja procedury określa wartość odwołania funkcyjnego.

3.2.4. Funkcje standardowe.

Poniższe identyfikatory są nazwami funkcji standardowych ALGOLU, pod warunkiem, że w programie nie ma deklaracji odnoszących się do nich /patrz rozdział 5. "Deklaracje"/:

ABS

ARCTAN

COS

ENTIER

EXP

INCHAR

LN

SIGN

SIN

SQRT

Jeżeli programista nie chce korzystać w programie lub jego części z jakiejś funkcji standardowej, to może on używać jej nazwy jako identyfikatora zadeklarowanej przez siebie wielkości.

3.2.4.1. Znaczenie.

Poniższe odwołania do funkcji standardowych oznaczają odpowiednio:

ABS(E) wartość bezwzględną wartości wyrażenia E;

ARCTAN(E) wartość główną funkcji arcus tangens E;

COS(E) wartość funkcji cosinus E;

ENTIER(E) wartość największej liczby całkowitej nie większej od wartości E;

EXP(E) wartość funkcji wykładniczej wartości E;

INCHAR(E) wartość symbolu wprowadzonego z urządzenia wejściowego związanego z numerem symbolicznej operacji wejścia E;

LN(E) wartość logarytmu naturalnego wyrażenia E;

SIGN(E)	znak algebraiczny wartości E (+1 dla E>0, 0 dla E=0, -1 dla E<0);
SIN(E)	wartość funkcji sinus E;
SQRT(E)	wartość dodatniego pierwiastka kwadratowego z E.

Funkcje te są określone zarówno dla argumentów typu REAL, jak i typu INTEGER. Funkcje: ENTIER, SIGN i INCHAR mają wartości typu INTEGER, a pozostałe funkcje - wartości typu REAL. Funkcje: ARCTAN, COS, EXP, LN, SIN i SQRT - są obliczane za pomocą algorytmów dających wyniki z błędem bezwzględnym mniejszym niż 2^{-38}.

Wywołanie funkcji standardowych LN i SQRT dla ujemnego argumentu powoduje przerwanie wykonywania programu. Obliczenie wartości LN(0) nie powoduje przerwania programu, lecz daje w wyniku najmniejszą liczbę ujemną maszyny z nadmiarem.

3.3. Wyrażenia arytmetyczne.

3.3.1. Składnia.

⟨operator typu dodawania⟩ ::= +|-

⟨operator typu mnożenia⟩ ::=*| /| DIV

⟨wyrażenie pierwotne⟩ ::= ⟨liczba bez znaku⟩|⟨zmienna⟩|
⟨odwołanie funkcyjne⟩|
(⟨wyrażenie arytmetyczne⟩)

⟨czynnik⟩ ::= ⟨wyrażenie pierwotne⟩|⟨czynnik⟩ POWER
⟨wyrażenie pierwotne⟩

⟨składnik⟩ ::= ⟨czynnik⟩|⟨składnik⟩
⟨operator typu mnożenia⟩ ⟨czynnik⟩

⟨proste wyrażenie arytmetyczne⟩ ::= ⟨składnik⟩|
⟨operator typu dodawania⟩
⟨składnik⟩|⟨proste
wyrażenie arytmetyczne⟩
⟨operator typu dodawania⟩
⟨składnik⟩

⟨warunek⟩ ::= IF ⟨wyrażenie boolowskie⟩ THEN

⟨wyrażenie arytmetyczne⟩ ::= ⟨proste wyrażenie arytmetyczne⟩|
⟨warunek⟩
⟨proste wyrażenie arytmetyczne⟩
ELSE ⟨wyrażenie arytmetyczne⟩

3.3.2. Przykłady.

Wyrażenia pierwotne:

7.394E-8

SUM

W[I+2,8]

COS (Y+Z*3)

(A-3/Y+VU POWER 8)

Czynniki:

OMEGA

SUM POWER COS(Y+Z*3)

7.394E-8 POWER W[I+2,8] POWER (A-3/Y+VU POWER 8)

Składniki:

```
U
OMEGA*SUM POWER COS(Y+Z*3)/7.394E-8 POWER
   W[I+2,8] POWER (A-3/Y+VU POWER 8)
```

Proste wyrażenie arytmetyczne:

```
U-YU + OMEGA*SUM POWER COS(Y+Z*3)/7.394E-8
   POWER W[I+2,8] POWER (A-3/Y+VU POWER 8)
```

Wyrażenia arytmetyczne:

```
W*U - Q[S+CU] POWER 2
IF Q GREATER 0 THEN S+3*Q/A ELSE 2*S+3*Q
IF A LESS 0 THEN U+V
  ELSE IF A*B  GREATER  17  THEN U/V
       ELSE IF K NOTEQUAL Y THEN V/U ELSE 0
A*SIN(OMEGA*T)
0.57E12*A[N*(N-1) /2,0]
(A*ARCTAN(Y)+Z) POWER (7+Q)
IF Q THEN N-1 ELSE N
IF A LESS 0 THEN A/B ELSE IF B = 0 THEN B/A ELSE Z
```

3.3.3. Znaczenie.

Wyrażenie arytmetyczne jest regułą na obliczanie wartości liczbowej. W przypadku prostego wyrażenia arytmetycznego wartość tę otrzymujemy wykonując wskazane operacje arytmetyczne na aktualnych liczbowych wartościach wyrażeń pierwotnych /szczegóły podane są w punkcie 3.3.4./.
Aktualna wartość liczbowa wyrażenia pierwotnego jest oczywista, jeżeli wyrażeniem tym jest liczba. Dla zmiennych jest

to wartość bieżąca nadana ostatnim przypisaniem, a dla odwołań funkcyjnych jest to wartość otrzymana w wyniku wykonania obliczeń według reguł określających procedurę, zastosowanych do bieżących wartości parametrów procedury, danych w wyrażeniu /patrz punkt 5.4.4. "Wartości funkcji"/. Wartość wyrażenia arytmetycznego ujętego w nawiasy należy wyrazić rekurencyjnie przez wartości pozostałych trzech typów wyrażeń pierwotnych.

W bardziej ogólnym wyrażeniu arytmetycznym, zawierającym warunki, wybiera się jedno z prostych wyrażeń arytmetycznych, zgodnie z aktualnymi wartościami wyrażeń boolowskich /patrz punkt 3.4. "Wyrażenia boolowskie"/. Wyboru dokonuje się w sposób następujący:
wartości wyrażeń boolowskich, występujących w warunkach, obliczamy kolejno dotąd, dopóki nie znajdziemy wyrażenia o wartości TRUE. Wówczas wartością wyrażenia arytmetycznego będzie wartość pierwszego wyrażenia arytmetycznego, następującego po tym wyrażeniu boolowskim, konstrukcja

ELSE ⟨proste wyrażenie arytmetyczne⟩

jest równoważna konstrukcji

ELSE IF TRUE THEN ⟨proste wyrażenie arytmetyczne⟩.

3.3.4. Operatory i typy.

Poza wyrażeniami boolowskimi występującymi w warunkach, części składowe prostych wyrażeń arytmetycznych muszą być typu REAL lub INTEGER patrz /punkt 5.1. "Deklaracje zmiennych prostych"/. Znaczenie podstawowych operatorów i typy

wyrażeń, do których one prowadzą, określają następujące reguły:

3.3.4.1. Operatory +, - i* mają zwykły sens (dodawanie, odejmowanie i mnożenie).
Wyrażenie będzie typu INTEGER, jeżeli oba jego operandy są typu INTEGER, w przeciwnym przypadku będzie ono typu REAL.

3.3.4.2. Operacje ⟨składnik⟩ / ⟨czynnik⟩ oraz ⟨składnik⟩ DIV ⟨czynnik⟩ oznaczają dzielenie, rozumiane jako mnożenie składnika przez odwrotność czynnika, z uwzględnieniem odpowiednich reguł pierwszeństwa /patrz punkt 3.3.5/. Tak więc na przykład:
A / B*7 / (P - Q)* V / S
oznacza w zapisie konwencjonalnym:
$$((((A.(B^{-1})) \cdot 7) \cdot ((P - Q)^{-1})) \cdot V) \cdot (S^{-1})$$
Operator "/" jest określony dla wszystkich czterech kombinacji typów REAL i INTEGER składnika oraz czynnika, i w każdym przypadku daje wynik REAL.
Operator DIV jest określony tylko dla obu argumentów typu INTEGER i daje wynik typu INTEGER, określony matematycznie wzorem:
A DIV B = SIGN(A/B)* ENTIER (ABS(A/B))
/patrz punkty 3.2.4 oraz 3.2.5/.

3.3.4.3. Operacja ⟨czynnik⟩ POWER ⟨wyrażenie pierwotne⟩ oznacza potęgowanie, przy czym czynnik jest podsta-

wą, a wyrażenie pierwotne wykładnikiem. Tak więc na przykład:

2 POWER N POWER K oznacza $(2^N)^K$

natomiast

2 POWER (N POWER M) oznacza $(2^N)^M$.

Oznaczmy literą I liczby typu INTEGER, literą R - liczby typu REAL, literą A - liczby typu REAL lub INTEGER. Wtedy wynik operacji potęgowania określony jest następującymi regułami:

A POWER I: jeżeli I>0, to wynik równa się A*A*...*A (I razy) i jest tego samego typu, co A;

jeżeli I = 0 i jeżeli A ≠ 0, to wynik równa się 1 i jest tego samego typu co A;

jeżeli A = 0, to wynik jest nieokreślony;

jeżeli I<0 i jeżeli A ≠ 0, to wynik równa się 1/(A* A*...*A) (-I razy) i jest typu REAL;

jeżeli A = 0, to wynik jest nieokreślony;

A POWER R: jeżeli A>0, to wynik wyliczany jest wzorem EXP(R*LN(A)) i jest typu REAL;

jeżeli A = 0 i jeżeli R>0, to wynik równa się 0.0 i jest typu REAL;
jeżeli R≤0, to wynik jest nieokreślony;
jeżeli A<0, to wynik jest nieokreślony.

3.3.5. Pierwszeństwo operatorów.

Operacje w wyrażeniu wykonuje się na ogół od lewej do prawej, jednak z uwzględnieniem następujących dodatkowych reguł:

3.3.5.1. Zgodnie ze składnią opisaną w punkcie 3.3.1. pierwszeństwo operatorów jest następujące:

pierwszy: POWER
drugie : * / DIV
trzecie : + -

3.3.5.2. Wyrażenie ujęte w nawiasy: otwierający i odpowiadający mu zamykający, oblicza się niezależnie od innych, a jego wartości używa się w dalszych obliczeniach. Dlatego w wyrażeniu można zawsze osiągnąć dowolny porządek wykonywania operacji przez odpowiednie rozstawienie nawiasów.

3.3.6. Arytmetyka wielkości typu REAL i INTEGER

Zarówno zmienne typu REAL jak i typu INTEGER mają w maszynie tę samą reprezentację zmiennoprzecinkową. W związku z tym wartości zmiennych muszą zawierać się odpowiednio

w przedziałach:

-2^{38} = -274877906944 wartość zmiennej typu INTEGER

274877906944 = 2^{38}

2^{-256} = 0.879E-77 ABS wartość zmiennej typu REAL

0.5689E77 = 2^{255}

Jeżeli wartość wyrażenia, które zgodnie z regułami podanymi w punkcie 3.3.4. jest typu INTEGER, nie mieści się w podanym wyżej przedziale, to wynik obliczenia będzie niedokładny.

3.4. Wyrażenie boolowskie.

3.4.1. Składnia.

⟨relacja⟩ ::=⟨proste wyrażenie arytmetyczne⟩
⟨operator relacji⟩
⟨proste wyrażenie arytmetyczne⟩

⟨pierwotne wyrażenie boolowskie⟩ ::=⟨wartość logiczna⟩|
⟨zmienna⟩|
⟨odwołanie funkcyjne⟩|
⟨relacja⟩|
(⟨wyrażenie boolowskie⟩)

⟨wtórne wyrażenie boolowskie⟩ ::=⟨pierwotne wyrażenie boolowskie⟩|
NOT⟨pierwotne wyrażenie boolowskie⟩

⟨czynnik boolowski⟩::=⟨wtórne wyrażenie boolowskie⟩|⟨czynnik boolowski⟩

AND⟨wtórne wyrażenie boolowskie⟩

⟨składnik boolowski⟩ ::=⟨czynnik boolowski⟩|

⟨składnik boolowski⟩

OR⟨czynnik boolowski⟩

⟨implikacja⟩ ::=⟨składnik boolowski⟩ |⟨implikacja⟩

IMPL⟨składnik boolowski⟩

⟨proste wyrażenie boolowskie⟩ ::=⟨implikacja⟩|⟨proste

wyrażenie boolowskie⟩

EQUIV ⟨ implikacja⟩

⟨wyrażenie boolowskie⟩ ::=⟨proste wyrażenie boolowskie⟩|

⟨warunek⟩

⟨proste wyrażenie boolowskie⟩ ELSE

⟨wyrażenie boolowskie⟩

3.4.2. Przykłady.

```
X = -2
X EQUAL -2
Y GREATER V OR Z LESS Q
A + B GREATER -5 AND Z - D GREATER Q POWER 2
P AND Q OR X NOTEQUAL Y
Q EQUIV NOT A AND B AND NOT C OR D OR E IMPL NOT F
IF K LESS I THEN S GREATER W ELSE H NOTGREATER C
IF IF IF A THEN B ELSE C THEN D ELSE F THEN G
                                    ELSE H LESS K
```

3.4.3. Znaczenie.

Wyrażenie boolowskie jest regułą na obliczanie wartości logicznej. Zasady obliczeń są tu analogiczne do zasad odno-

szących się do wyrażeń arytmetycznych, podanych w punkcie 3.3.3.

3.4.4. Typy.

Zmiennym i odwołaniom funkcyjnym, użytym jako pierwotne wyrażenie boolowskie, należy przypisać typ BOOLEAN /patrz punkty 5.1. "Deklaracje zmiennych prostych" oraz 5.4.4. "Wartości funkcji"/.

3.4.5. Operatory.

Relacja ma wartość TRUE, jeżeli jest spełniona dla wyrażeń, wchodzących w jej skład. W przeciwnym razie relacja ma wartość FALSE. Poniższa tabelka określa interpretację operatorów logicznych: NOT (nie), AND (i), OR (lub), IMPL (implikuje), EQUIV (równoważne):

B1	FALSE	FALSE	TRUE	TRUE
B2	FALSE	TRUE	FALSE	TRUE
NOT B1	TRUE	TRUE	FALSE	FALSE
B1 AND B2	FALSE	FALSE	FALSE	TRUE
B1 OR B2	FALSE	TRUE	TRUE	TRUE
B1 IMPL B2	TRUE	TRUE	FALSE	TRUE
B1 EQUIV B2	TRUE	FALSE	FALSE	TRUE

3.4.6. Pierwszeństwo operatorów.

Operacje w wyrażeniu wykonuje się na ogół od lewej do prawej, jednak z uwzględnieniem następujących dodatkowych reguł:

3.4.6.1. Zgodnie ze składnią opisaną w punkcie 3.4.1. operacje wykonywane są w następującej kolejności:

pierwsze: wyrażenia arytmetyczne, zgodnie z punktem 3.3.5.

drugie : LESS,NOTGREATER,=,EQUAL,NOTLESS, GREATER,NOTEQUAL

trzecie : NOT

czwarte : AND

piąte : OR

szóste : IMPL

siódme : EQUIV

3.4.6.2. Zastosowanie nawiasów interpretuje się w sensie podanym w punkcie 3.3.5.2.

3.5. Wyrażenia sterujące.

3.5.1. Składnia.

⟨etykieta⟩ ::= ⟨identyfikator⟩

⟨identyfikator przełącznika⟩ ::= ⟨identyfikator⟩

⟨przełączenie⟩ ::= ⟨identyfikator przełącznika⟩ [⟨wyrażenie indeksowe⟩]

⟨proste wyrażenie sterujące⟩ ::= ⟨etykieta⟩ | ⟨przełączenie⟩ | (⟨wyrażenie sterujące⟩)

⟨wyrażenie sterujące⟩ ::= ⟨proste wyrażenie sterujące⟩ | ⟨warunek⟩ ⟨proste wyrażenie sterujące⟩ ELSE ⟨wyrażenie sterujące⟩

3.5.2. Przykłady.

```
E3
P9
WARIANT[N-1]
DROGA[IF Y LESS 0 THEN N ELSE N+1]
IF AB LESS C THEN E3 ELSE Q[IF W NOTGREATER 0 THEN
                                P9 ELSE N]
```

3.5.3. Znaczenie.

Wyrażenie sterujące jest regułą na określenie etykiety instrukcji /patrz rozdział 4. "Instrukcje"/. Reguły obliczania wartości wyrażenia sterującego są analogiczne do reguł podanych dla wyrażenia arytmetycznego /patrz punkt 3.3.3./. Wyrażenia boolowskie zawarte w warunkach wyznaczają jedno z prostych wyrażeń sterujących. Jeżeli jest ono etykietą, to żądany wynik już otrzymano. Przełączenie odsyła do deklaracji odpowiedniego przełącznika /patrz punkt 5.3. "Deklaracje przełączników"/ i według aktualnej wartości liczbowej jego wyrażenia indeksowego wybiera jedno z wyrażeń sterujących umieszczonych na liście w deklaracji przełącznika, licząc te wyrażenia od lewego do prawego. W związku z tym, że wybrane w ten sposób wyrażenie sterujące może znów okazać się przełączeniem, proces obliczania wartości jest na ogół rekurencyjny.

3.5.4. Wyrażenie indeksowe.

Obliczanie wartości wyrażenia indeksowego przebiega tak

samo, jak w przypadku zmiennej indeksowanej /patrz punkt 3.1.4.2 /. Wartość przełączenia jest określona tylko wtedy, gdy wyrażenie indeksowe ma jedną z wartości dodatnich 1,2,...,N, gdzie N jest liczbą pozycji na liście w deklaracji przełącznika.

4. INSTRUKCJE.

Jednostki operacyjne języka nazywają się instrukcjami. Instrukcje wykonuje się zwykle w takim porządku, w jakim są napisane. Porządek ten mogą zmienić jedynie instrukcje skoku, jawnie określające swój następnik oraz instrukcje warunkowe, które mogą spowodować pominięcie pewnych instrukcji.

Instrukcje można opatrywać etykietami, co pozwala określić szczególne dynamiczne następstwo instrukcji.

W związku z tym, że ciągi instrukcji można grupować w instrukcje złożone i bloki, z konieczności definicja instrukcji jest rekurencyjna. W składniowej definicji instrukcji zakłada się, że deklaracje /patrz rozdział 5/, są już określone, stanowią one bowiem istotną część struktury składniowej.

4.1. Instrukcje złożone i bloki.

4.1.1. Składnia.

⟨instrukcja podstawowa bez etykiety⟩ ::=⟨instrukcja przypisania⟩|
⟨instrukcja skoku⟩|
⟨instrukcja pusta⟩|
⟨instrukcja procedury⟩

⟨instrukcja podstawowa⟩ ::=⟨instrukcja podstawowa bez etykiety⟩|
⟨etykieta⟩:⟨instrukcja podstawowa⟩

⟨instrukcja bezwarunkowa⟩ ::= ⟨instrukcja podstawowa⟩ |
⟨instrukcja złożona⟩ | ⟨blok⟩

⟨instrukcja⟩ ::= ⟨instrukcja bezwarunkowa⟩ |
⟨instrukcja warunkowa⟩ | ⟨instrukcja cyklu⟩

⟨koniec instrukcji złożonej⟩ ::= ⟨instrukcja⟩ END | ⟨instrukcja⟩ ;
⟨koniec instrukcji złożonej⟩

⟨początek bloku⟩ ::= BEGIN ⟨deklaracja⟩ | ⟨początek bloku⟩ ;
⟨deklaracja⟩

⟨instrukcja złożona bez etykiety⟩ ::= BEGIN ⟨koniec
instrukcji złożonej⟩

⟨blok bez etykiety⟩ ::= ⟨początek bloku⟩ ; ⟨koniec instrukcji
złożonej⟩

⟨instrukcja złożona⟩ ::= ⟨instrukcja złożona bez etykiety⟩ |
⟨etykieta⟩ : ⟨instrukcja złożona⟩

⟨blok⟩ ::= ⟨blok bez etykiety⟩ | ⟨etykieta⟩ : ⟨blok⟩

⟨program⟩ ::= ⟨blok⟩ | ⟨instrukcja złożona⟩

Dla wyjaśnienia tej składni oznaczmy dowolne instrukcje, deklaracje i etykiety odpowiednio literami: I, D i E. Wtedy podstawowe jednostki składniowe będą miały następującą postać:

Instrukcja złożona:

E: E: ... BEGIN I; I; ... I; I END

Blok:

E: E: ... BEGIN D; D; ... D; I; ... I; I END

Należy przy tym pamiętać, że każda z instrukcji I sama

może być instrukcją złożoną lub blokiem.

4.1.2. Przykłady.

Instrukcje podstawowe:

```
A := P + Q
GO TO KONIEC
START: DALSZYCIAG: W:= 7.993
```

Instrukcja złożona:

```
BEGIN X:=0; FOR Y:=1 STEP 1 UNTIL N DO X:=X+A[Y];
   IF X GREATER Q THEN GOTO STOP ELSE IF X GREATER
                              W-2 THEN GO TO S;
  AW: ST: W:=X + BOB
END
```

Blok:

```
     Q: BEGIN INTEGER I,K; REAL W;
              FOR I:=1 STEP 1 UNTIL M DO
                FOR K:=I+1 STEP 1 UNTIL M DO
                  BEGIN W:= A[I,K];
                        A[I,K]:= A[K,I];
                        A[K,I]:= W
                  END DLA I ORAZ K
                END KONIEC BLOKU Q
```

4.1.3. Znaczenie.

Każdy blok wprowadza automatycznie nowy poziom oznaczeń. Realizuje się to w ten sposób, że dowolny identyfikator występujący wewnątrz bloku może być odpowiednią deklaracją

/rozdział 5. "Deklaracje"/ uczyniony lokalnym w tym bloku. Oznacza to: /a/ że obiekt, reprezentowany przez ten identyfikator wewnątrz danego bloku, nie istnieje poza tym blokiem oraz /b/ że dowolny obiekt, reprezentowany przez ten sam identyfikator poza danym blokiem, jest całkowicie niedostępny w tym bloku.

Identyfikatory /z wyjątkiem tych, które oznaczają etykiety/ spotykane wewnątrz bloku i nie zadeklarowane w nim, nie są w nim lokalne, tj. reprezentują te same obiekty wewnątrz danego bloku, co i w bloku pezpośrednio obejmującym dany blok. Etykietę oddzieloną dwukropkiem od instrukcji, a więc przyporządkowaną tej instrukcji, traktuje się tak, jak gdyby była ona zadeklarowana na początku obejmującego bloku, tj. w najmniejszym bloku, którego nawiasy BEGIN i END zawierają wspomnianą instrukcję. W tym kontekście treść procedury należy traktować tak jak gdyby była ona ujęta w nawiasy BEGIN i END i stanowiła blok.

W związku z tym, że instrukcja w bloku sama może być blokiem, pojęcia "lokalny" lub "nielokalny" należy rozumieć rekurencyjnie.
Tak więc identyfikator nielokalny w bloku A może być lokalny lub nielokalny w bloku B, w którym A jest jedną z instrukcji.

Liczba poziomów bloków, które mogą być deklarowane jeden wewnątrz drugiego została ograniczona do 63.

4.2. Instrukcje przypisania.

4.2.1. Składnia.

⟨lewa strona⟩::=⟨zmienna⟩:=|⟨identyfikator procedury⟩:=

⟨lista lewych stron⟩::=⟨lewa strona⟩|⟨lista lewych stron⟩ ⟨lewa strona⟩

⟨instrukcja przypisania⟩ ::=⟨lista lewych stron⟩ ⟨wyrażenie arytmetyczne⟩|
⟨lista lewych stron⟩ ⟨wyrażenie boolowskie⟩

4.2.2. Przykłady.

```
S:= P[0]:= N:= N + 1 + S
N:= N + 1
A:= B/C - V - Q×S
S[V,K + 2]:= 3 - ARCTAN(S×ZETA)
V:= Q GREATER Y AND Z
```

4.2.3. Znaczenie.

Instrukcje przypisania służą do przypisania wartości wyrażenia jednej lub wielu zmiennym lub identyfikatorom procedury. Przypisanie wartości identyfikatorom procedury może mieć miejsce jedynie w treści procedury określającej wartość funkcji /punkt 5.4.4 /.

Proces przypisania wartości przebiega w trzech następujących krokach:

4.2.3.1. Wszystkie wyrażenia indeksowe występujące w zmiennych lewych stron oblicza się kolejno od lewego do

prawego.

4.2.3.2. Oblicza się wartość wyrażenia.

4.2.3.3. Wartość wyrażenia przypisuje się wszystkim zmiennym lewych stron, w których wyrażenia indeksowe mają takie wartości jakie zostały obliczone w kroku 4.2.3.1.

4.2.4. Typy.

Wszystkie zmienne i identyfikatory procedury na liście lewych stron powinny być jednakowego typu. Jeżeli jest to typ BOOLEAN, to wyrażenie także musi być typu BOOLEAN. Jeżeli jest to typ REAL lub INTEGER, to wyrażenie musi być arytmetyczne. Jeżeli typ wyrażenia arytmetycznego różni się od typu związanych z nim zmiennych i identyfikatorów procedury, to automatycznie wykonywane jest odpowiednie przekształcenie. Przy przekształceniu wartości wyrażenia E z typu REAL na typ INTEGER otrzymuje się wynik:

ENTIER(E + 0.5)

Typ związany z identyfikatorem procedury jest określony przez deklarator, będący pierwszym symbolem w deklaracji tej procedury /punkt 5.4.4./.

4.3. Instrukcje skoku.

4.3.1. Składnia.

⟨instrukcja skoku⟩::= GO TO⟨wyrażenie sterujące⟩|
GOTO⟨wyrażenie sterujące⟩

4.3.2. Przykłady.

```
GO TO E3
GOTO WARIANT[N-1]
GOTO DROGA [IF Y LESS 0 THEN N ELSE N+1]
GO TO IF AB LESS C THEN E3 ELSE Q[IF W NOTGREATER
                                  0 THEN P9 ELSE N]
```

4.3.3. Znaczenie.

Instrukcja skoku przerywa naturalny porządek wykonywania instrukcji, zgodny z porządkiem, w jakim je napisano. Instrukcja skoku wyznacza swój następnik przez wartość wyrażenia sterującego. Tak więc następną wykonywaną instrukcją będzie ta, dla której wspomniana wartość jest etykietą.

4.3.4. Ograniczenie.

W związku z tym, że etykiety są z natury lokalne, żadna instrukcja skoku nie może prowadzić z zewnątrz bloku do jego wnętrza. Instrukcja skoku może jednak prowadzić z zewnątrz do instrukcji złożonej.

4.3.5. Skok przy nieokreślonym przełączeniu.

Jeżeli wyrażenie sterujące jest przełączeniem o nieokreślonej wartości, to instrukcja skoku jest równoważna instrukcji pustej.

4.4. Instrukcje puste.

4.4.1. Składnia.

⟨instrukcja pusta⟩ ::= ⟨puste⟩

4.4.2. Przykłady.

```
L: BEGIN ... ; ALFA: END
```

4.4.3. Znaczenie.

Instrukcja pusta nie wykonuje żadnej czynności. Można ją wykorzystać do umieszczenia etykiety.

4.5. Instrukcje warunkowe.

4.5.1. Składnia.

⟨warunek⟩ ::= IF⟨wyrażenie boolowskie⟩ THEN

⟨instrukcja warunkowa niepełna⟩ ::= ⟨warunek⟩ ⟨instrukcja bezwarunkowa⟩

⟨instrukcja warunkowa⟩ ::= ⟨instrukcja warunkowa niepełna⟩|
⟨instrukcja warunkowa niepełna⟩
ELSE ⟨instrukcja⟩|⟨warunek⟩
⟨instrukcja cyklu⟩|⟨etykieta⟩ :
⟨instrukcja warunkowa⟩

4.5.2. Przykłady.

```
IF X GREATER 0 THEN N:=N+1
IF V GREATER U THEN V:=N+M ELSE GOTO R
IF S LESS 0 OR P NOTGREATER Q
   THEN AA: BEGIN IF Q LESS V THEN A:=V/S ELSE
                                  Y:=2*A END
ELSE IF V GREATER S THEN A:=V-Q
                    ELSE IF V GREATER S-1 THEN GO TO SA1
```

4.5.3. Znaczenie.

Instrukcje warunkowe powodują pominięcie lub wykonanie

pewnych instrukcji zależnie od bieżących wartości określonych wyrażeń boolowskich .

4.5.3.1. Instrukcja warunkowa niepełna.

Instrukcja bezwarunkowa występująca w instrukcji warunkowej niepełnej będzie wykonana, jeżeli wyrażenie boolowskie stanowiące część warunku ma wartość TRUE; w przeciwnym razie ta instrukcja zostanie pominięta i dalsze działanie rozpocznie się od następnej instrukcji.

4.5.3.2. Instrukcje warunkowe.

Zgodnie ze składnią są możliwe dwa różne rodzaje instrukcji warunkowych. Można je zilustrować następującymi przykładami:

```
IF B1 THEN I1 ELSE IF B2 THEN I2 ELSE I3; I4
IF B1 THEN I1 ELSE IF B2 THEN I2 ELSE IF B3
                                THEN I3; I4
```

B1, B2 i B3 są tu wyrażeniami boolowskimi, I1, I2 i I3 są instrukcjami bezwarunkowymi, a I4 instrukcją następującą po instrukcji warunkowej.

Wykonanie instrukcji warunkowej można opisać następująco:
wartość wyrażeń boolowskich w warunkach oblicza się kolejno od lewego do prawego, aż do znalezienia wartości TRUE; następnie wykonuje się instrukcję bezwarunkową napisaną bezpośrednio po tym wyrażeniu; jeżeli ta instrukcja nie określa sama

swojego następnika, to będzie nim I4, tj. instrukcja następująca po pełnej instrukcji warunkowej; tak więc działanie ogranicznika ELSE polega na tym, że na następnik instrukcji, po której stoi ELSE, wyznacza on instrukcję napisaną po pełnej instrukcji warunkowej.

Konstrukcja

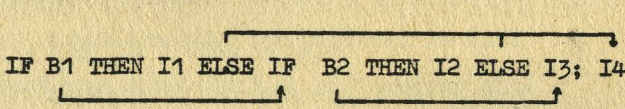
ELSE ⟨instrukcja bezwarunkowa⟩

jest równoważna konstrukcji

ELSE IF TRUE THEN ⟨instrukcja bezwarunkowa⟩

Jeżeli żadne z wyrażeń boolowskich występujących w warunkach nie ma wartości TRUE, to wynik wykonania takiej instrukcji warunkowej będzie równoważny wynikowi wykonania instrukcji pustej.

Dla dalszych wyjaśnień może być pożyteczny następujący schemat:



4.5.4. Skok do wnętrza instrukcji warunkowej.

Wynik wykonania instrukcji skoku, prowadzącej do wnętrza instrukcji warunkowej, wypływa bezpośrednio z wyjaśnionego wyżej działania ogranicznika ELSE.

4.6. Instrukcje cyklu.

4.6.1. Składnia.

⟨element listy cyklu⟩ ::= ⟨wyrażenie arytmetyczne⟩|
⟨wyrażenie arytmetyczne⟩ STEP
⟨wyrażenie arytmetyczne⟩ UNTIL
⟨wyrażenie arytmetyczne⟩|
⟨wyrażenie arytmetyczne⟩ WHILE
⟨wyrażenie boolowskie⟩

⟨lista cyklu⟩ ::= ⟨element listy cyklu⟩|⟨lista cyklu⟩ ,
⟨element listy cyklu⟩

⟨warunek cyklu⟩ ::= FOR ⟨zmienna⟩ := ⟨lista cyklu⟩ DO

⟨instrukcja cyklu⟩ ::= ⟨warunek cyklu⟩ ⟨instrukcja⟩|
⟨etykieta⟩:⟨instrukcja cyklu⟩

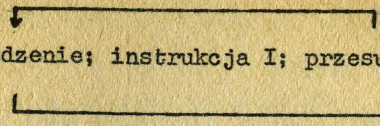
4.6.2. Przykłady.

```
FOR Q := 1 STEP S UNTIL N DO A[Q]:=B[Q]
FOR K := 1, V1*2 WHILE V1 LESS N DO
FOR J:= I+G,L, 1 STEP 1 UNTIL N,C+D DO A[K,J]:=B[K,J]
```

4.6.3. Znaczenie.

Warunek cyklu powoduje kolejne wykonanie następującej po nim instrukcji I zero lub więcej razy. Ponadto warunek cyklu przypisuje sterowanej przez siebie zmiennej kolejne wartości.

Proces ten można wyjaśnić za pomocą następującego schematu:



W tym schemacie słowo "inicjalizacja" oznacza wykonanie pierwszego przypisania zgodnie z warunkiem cyklu. "Przesunięcie" oznacza następne przypisanie według tego warunku. "Sprawdzenie" oznacza badanie czy wykonano ostatnie przypisanie: jeżeli tak, to będzie wykonany następnik instrukcji cyklu; w przeciwnym przypadku wykonuje się instrukcję napisaną po warunku cyklu.

4.6.4. Elementy listy cyklu.

Lista cyklu jest regułą według której otrzymywane są wartości, przypisywane kolejno zmiennej sterowanej przez warunek cyklu. Wartości te otrzymuje się z kolejnych elementów listy cyklu. Ciąg wartości generowany przez każdy z trzech składniowo możliwych rodzajów elementów listy cyklu oraz odpowiadające mu wykonania instrukcji cyklu - określają następujące reguły:

4.6.4.1. Wyrażenie arytmetyczne.

Taki element daje tylko jedną wartość, a mianowicie wartość wyrażenia arytmetycznego, obliczoną bezpośrednio przed odpowiadający temu elementów wykonaniem instrukcji I.

4.6.4.2. Element postaci postępu arytmetycznego.

Element postaci A STEP B UNTIL C, gdzie A,B i C są wyrażeniami arytmetycznymi, określa taki porządek wykonywania, który można za pomocą innych instrukcji ALGOLu opisać tak:

```
V:=A;
L1: IF (V - C)*SIGN(B) GREATER 0 THEN GO TO ELE-
                                    MENTWYCZERPANO;
instrukcja I;
V:=V + B;
GO TO L1;
```

gdzie V jest zmienną sterowaną przez warunek cyklu, a etykieta ELEMENTWYCZERPANO prowadzi do obliczenia następnego elementu listy cyklu lub - jeżeli ten element jest ostatni na liście - do następnej instrukcji programu.

4.6.4.3. Element "podczas gdy".

Wykonanie sterowane przez element listy cyklu postaci E WHILE F, gdzie E jest wyrażeniem arytmetycznym, a F - boolowskim, można za pomocą innych instrukcji ALGOLu opisać tak:

```
L3: V:=E;
    IF NOT F THEN GO TO ELEMENTWYCZERPANO;
    instrukcja I;
    GO TO L3;
```

Oznaczenia - jak w punkcie 4.6.4.2.

4.6.5. Końcowa wartość zmiennej sterowanej przez warunek cyklu.

Tożsamość zmiennej sterowanej przez warunek cyklu nie jest ustalana raz na początku każdej aktywizacji pętli i może zależeć od wyniku obliczenia wyrażeń zmieniających wartość zmiennej indeksowanej, zawartych w instrukcji sterowanej. Po wyjściu z instrukcji I /jeżeli jest ona instrukcją złożoną/ za pomocą instrukcji skoku wartość zmiennej sterowanej przez warunek cyklu będzie taka, jak bezpośrednio przed wykonaniem instrukcji skoku. Jeżeli natomiast wyjście z instrukcji I nastąpiło na skutek wyczerpania listy cyklu, to ostatnia wartość zmiennej sterowanej nie jest określona.

4.6.6. Skok do wnętrza instrukcji cyklu.

Wynik wykonania instrukcji skoku znajdującej się na zewnątrz instrukcji cyklu, a odnoszącej się do etykiety wewnątrz tej instrukcji cyklu, nie jest określony.

4.7. Instrukcje procedury.

4.7.1. Składnia.

⟨Instrukcja procedury⟩ ::= ⟨identyfikator procedury⟩ ⟨zbiór parametrów aktualnych⟩

4.7.2. Przykłady.

```
TRANSPOZYCJA (W) STOPIEŃ: (V+1)
MAX (A, N, M, YY, I,K)
ILOCZYNSKALARNY (A[T, P, U], B[P], 10,P,Y)
OUT (0, 'Y3B-6D', A[I], M+X/Q)
```

4.7.3. Znaczenie.

Instrukcja procedury powoduje wykonanie treści procedury /punkt 5.4. "Deklaracje procedur"/. Ponadto w ramach wykonania instrukcji procedury wykonywane są następujące czynności:

4.7.3.1. Przypisanie wartości /wywołanie przez wartość/.

Wszystkim parametrom formalnym, występującym w nagłówku procedury na liście parametrów wywoływanych przez wartość, przypisuje się wartości /punkt 2.8. "Wartości i typy"/ odpowiednich parametrów aktualnych. Przypisania te wykonywane są bezpośrednio przed przejściem do treści procedury. Wynik tych przypisań jest taki, jak gdyby treść procedury była objęta dodatkowym blokiem i w tym bloku zostało wykonane przypisanie wartości zmiennym lokalnym w nim, zgodne co do typu z odpowienimi specyfikacjami /punkt 5.4.5/. W rezultacie zmienne wywołane przez wartość traktuje się jako nielokalne w treści procedury, ale lokalne we wspomnianym fikcyjnym bloku /punkt 5.4.3/.

4.7.3.2. Zamiana nazwy /wywołanie przez nazwę/.

Każdy parametr formalny, nie wymieniony na liście parametrów wywoływanych przez wartość, zastępowany jest wszędzie w treści procedury odpowiednim parametrem aktualnym, ujętym /tam, gdzie jest to składniowo możliwe/ w nawiasy. Możliwe kolizje między identyfikatorami włączonymi przez taki proces do treści procedury a identyfikatorami, które występowały w niej uprzednio, usuwane są przez odpowiednią zmianę identyfikatory parametrow formalnych lub zmiennych

lokalnych.

4.7.3.3. Zamiana i wykonanie treści procedury.

Na koniec treść procedury zmodyfikowaną tak, jak opisano powyżej, umieszcza się w programie na miejscu instrukcji procedury i wykonuje. Jeżeli procedurę wywołano poza obszarem działania dowolnej wielkości nielokalnej w treści tej procedury, to kolizje między identyfikatorami włączonoymi do treści procedury przez proces zamiany a identyfikatorami, których deklaracje obowiązują w miejscu, w którym występuje instrukcja procedury lub odwołanie funkcyjne, usuwane są przez odpowienią zamianę identyfikatorów, których deklaracje obowiązują w tym miejscu.

4.7.4. Odpowiedniość między parametrami formalnymi i aktualnymi. Odpowiedniość między parametrami aktualnymi instrukcji procedury i parametrami formalnymi nagłówka procedury ustala się jak następuje: lista parametrów aktualnych instrukcji procedury powinna mieć tyle pozycji, ile ma lista parametrów formalnych nagłówka deklaracji prcedury; odpowiedniość ustanawiana jest przez zestawianie kolejnych par pozycji obu tych list.

4.7.5. Ograniczenia.

Na to, by instrukcja procedury była określana, potrzeba żeby działania na treści procedury, zdefiniowane w punktach 4.7.3. oraz 4.7.3.2. prowadziły do poprawnej instrukcji

w ALGOLu. Nakłada to na dowolną instrukcję procedury ograniczenia polegające na tym, że klasa i typ każdego parametru aktualnego powinny być zgodne z klasą i typem odpowiedniego parametru formalnego. Oto niektóre ważne przypadki tej reguły:

4.7.5.1. Jeżeli tekst jest parametrem aktualnym w instrukcji procedury różnej od standardowych instrukcji wyjścia /punkt 6.1. "Instrukcja wyjścia"/ lub w odwołaniu funkcyjnym, to ten tekst może być użyty w treści procedury jedynie jako parametr aktualny przy wywołaniu dalszych procedur. Ostatecznie, tekst może być użyty przez standardowe procedury wyjścia.

4.7.5.2. Parametrowi formalnemu, który w treści procedury jest lewą stroną pewnej instrukcji przypisania i w nagłówku procedury nie jest wymieniony na liście parametrów wywoływanych przez wartość, może odpowiadać tylko parametr aktualny będący zmienną /a więc szczególnym przypadkiem wyrażenia/.

4.7.5.3. Parametrowi formalnemu, który w treści procedury jest identyfikatorem macierzy, może odpowiadać tylko taki parametr aktualny, który jest identyfikatorem macierzy o tych samych wymiarach. Jeżeli parametr formalny jest wyszczególniony na liście parametrów wywoływanych przez wartość, to lokalna macierz, która pojawi się w treści procedury w wyniku przypisania, otrzyma te same granice indeksów, co i macierz aktualna.

4.7.5.4. Parametrowi formalnemu występującemu w nagłówku procedury na liście parametrów wywoływanych przez wartość, nie może odpowiadać identyfikator przełącznika ani procedury, ponieważ nie mają one wartości. Wyjątek stanowi identyfikator takiej procedury, której deklaracja zawiera zbiór pusty parametrów formalnych /punkt 5.4.1/ i określa wartość odwołania funkcyjnego /punkt 5.4.4./. Nazwa takiej procedury sama jest wyrażeniem.

4.7.5.5. Każdy parametr formalny może nakładać ograniczenia na typ związanego z nim parametru aktualnego /ograniczenia te są wymienione w nagłówku procedury w postaci specyfikacji/. W instrukcji procedury należy oczywiście przestrzegać tych ograniczeń.

4.7.6. Ograniczniki parametrów.

Wszystkie ograniczniki parametrów uważa się za równoważne. Nie ma żadnej zależności między ogranicznikami parametrów stosowanymi w instrukcji procedury i ogranicznikami występującymi w nagłówku procedury prócz żądania, by liczby tych ograniczników były jednakowe. Tak więc cała informacja, którą dają objaśniające ograniczniki, jest w istocie zbędna.

4.7.7. Ograniczenie.

Liczba parametrów formalnych procedury nie może przekraczać 127.

5. DEKLARACJE.

Za pomocą deklaracji określa się pewne własności wielkości używanych w programie i wiąże je z identyfikatorami. Deklaracja identyfikatora obowiązuje w jednym bloku. Poza tym blokiem używać tego identyfikatora można do innych celów /punkt 4.1.3/.

W sensie dynamicznym należy to rozumieć tak: od chwili wejścia do bloku /przez BEGIN, gdyż etykiety występujące wewnątrz są lokalne, a zatem niedostępne z zewnątrz/ wszystkie identyfikatory zadeklarowane w tym bloku mają znaczenie wynikające z natury podanych deklaracji. Jeżeli te identyfikatory były już określone za pomocą innych deklaracji poza danym blokiem, to na pewien czas nadaje się im nowy sens. Natomiast identyfikatory nie zadeklarowane w bloku zachowują dawne znaczenie. W chwili wyjścia z bloku /przez END lub instrukcję skoku/ wszystkie identyfikatory zadeklarowane w nim tracą swoje znaczenie lokalne.

Deklaracje można opatrzyć dodatkowym deklaratorem OWN. Skutek tego jest następujący: w chwili wejścia do bloku wartości wielkości typu OWN /będziemy mówili: "wielkości własnych"/ są takie, jakimi były w momencie ostatniego wyjścia z tego bloku, natomiast wartości wielkości, których deklaracje nie są opatrzone deklaratorem OWN, będą nieokreślone. Wszystkie identyfikatory programu poza etykietami i parametrami formalnymi z deklaracji procedur,

a także identyfikatorami funkcji standardowych i procedur standardowych /punkt 3.2.4 oraz rozdział 6/ muszą być zadeklarowane. Żadnego identyfikatora nie można deklarować na początku bloku więcej niż raz.

Składnia:

⟨deklaracja⟩ ::=⟨deklaracja zmiennych prostych⟩|
⟨deklaracja macierzy⟩|
⟨deklaracja przełącznika⟩|⟨deklaracja procedury⟩

5.1. Deklaracje zmiennych prostych.

5.1.1. Składnia.

⟨lista zmiennych prostych⟩ ::= ⟨zmienna prosta⟩|⟨zmienna prosta⟩,
⟨lista zmiennych prostych⟩

⟨typ⟩ ::= REAL | INTEGER | BOOLEAN

⟨typ lokalny lub własny⟩ ::= ⟨typ⟩| OWN ⟨typ⟩

⟨deklaracja zmiennych prostych⟩ ::= ⟨typ lokalny lub własny⟩
⟨lista zmiennych prostych⟩

5.1.2. Przykłady.

```
INTEGER P, Q, S
OWN BOOLEAN ACRYL, N
```

5.1.3. Znaczenie.

Deklaracja zmiennych prostych określa identyfikatory, które reprezentują zmienne proste i podaje ich typ. Zmienne, którym deklaracja nadaje typ REAL, mogą przyjmować wartości dodatnie, ujemne lub równe zeru. Zmienne, którym deklaracja nadaje typ INTEGER, mogą przyjmować wartości całkowite

dodatnie, ujemne lub równe zeru. Zmienne, którym deklaracja nadaje typ BOOLEAN, mogą przyjmować wartości TRUE lub FALSE. W wyrażeniach arytmetycznych w dowolnym miejscu, gdzie może wystąpić zmienna typu REAL, może również wystąpić zmienna typu INTEGER.

Znaczenie deklaratora OWN omówione zostało na początku tego rozdziału.

5.2. Deklaracje macierzy.

5.2.1. Składnia.

⟨ograniczenie dolne⟩ ::= ⟨wyrażenie arytmetyczne⟩

⟨ograniczenie górne⟩ ::= ⟨wyrażenie arytmetyczne⟩

⟨para ograniczeń⟩ ::= ⟨ograniczenie dolne⟩ :⟨ograniczenie górne⟩

⟨lista par ograniczeń⟩::=⟨para ograniczeń⟩|⟨lista par ograniczeń⟩,⟨para ograniczeń⟩

⟨segment macierzy⟩ ::= ⟨identyfikator macierzy⟩[lista par ograniczeń]|⟨identyfikator macierzy⟩ ,⟨segment macierzy⟩

⟨lista macierzy⟩ ::= ⟨segment macierzy⟩|⟨ lista macierzy⟩ , ⟨segment macierzy⟩

⟨deklaracja macierzy⟩ ::= ARRAY ⟨lista macierzy⟩|⟨typ lokalny lub własny⟩ ARRAY ⟨lista macierzy⟩

5.2.2. Przykłady.

```
ARRAY A, B, C[7:N,2:M] ,S[-2:10]
INTEGER ARRAY A[ IF  C  LESS 0 THEN 2 ELSE 1 : 20]
OWN REAL ARRAY Q[ -7:-1]
```

5.2.3. Znaczenie.

Deklaracja macierzy określa jeden lub wiele identyfikatorów reprezentujących wielowymiarowe macierze zmiennych indeksowanych. Deklaracja podaje też wymiary tych macierzy, ograniczenia indeksów i typy zmiennych.

5.2.3.1. Ograniczenia indeksów.

Ograniczenia indeksów dowolnej macierzy zadane są w postaci listy par ograniczeń w pierwszej parze nawiasów indeksowych występującej za identyfikatorem danej tablicy. Każda pozycja tej listy określa dolne i górne ograniczenie indeksu za pomocą dwóch wyrażeń arytmetycznych oddzielonych separatorem ":". Lista par ograniczeń podaje ograniczenia wszystkich indeksów w ich naturalnym porządku.

5.2.3.2. Wymiary.

Liczba wymiarów macierzy jest równa liczbie elementów na liście par ograniczeń.

5.2.3.3. Typy.

Wszystkie macierze opisane w jednej deklaracji są tego samego typu. Jeżeli brak deklaratora typu, to przyjmuje się, że macierze są typu REAL.

5.2.3.4. Segment macierzy.

Segment macierzy może określać nie więcej niż 255 macierzy.

5.2.4. Wyrażenia występujące jako ograniczenia indeksów.

5.2.4.1. Wyrażenia te wylicza się tak samo, jak wyrażenia indeksowe /punkt 3.1.4.2/.

5.2.4.2. Wyrażenia te mogą zależeć tylko od zmiennych i procedur nielokalnych w bloku, w którym obowiązuje deklaracja macierzy. Stąd wynika, że w najbardziej zewnętrznym bloku programu w deklaracjach macierzy mogą występować tylko stałe ograniczenia indeksów.

5.2.4.3. Pary ograniczeń dla macierzy, których deklaracje są poprzedzone deklaratorem OWN, muszą być liczbami całkowitymi. Oznacza to, że ograniczenia indeksów dla macierzy własnych nie mogą być dynamiczne.

5.2.4.4. Macierz jest określona tylko wtedy, gdy wartości wszytkich górnych ograniczeń indeksów nie są mniejsze od wartości odpowiednich dolnych ograniczeń.

5.2.4.5. Wyrażenia występujące jako ograniczenia indeksów oblicza się na nowo przy każdym wejściu do bloku.

5.2.5. Identyczność zmiennych indeksowanych.

Identyczność zmiennych indeksowanych nie ma związku z ograniczeniami indeksów podanymi w deklaracji macierzy. Jednak, nawet w przypadku gdy w deklaracji macierzy występuje deklarator OWN, wartości odpowiednich zmiennych indeksowanych są w dowolnej chwili określone tylko dla tych

indeksów, które są zawarte w ograniczeniach obliczonych ostatnim razem.

5.3. Deklaracja przełączników.

5.3.1. Składnia.

⟨lista przełączeń⟩ ::=⟨wyrażenie sterujące⟩|⟨ lista przełączeń⟩,⟨wyrażenie sterujące⟩

⟨deklaracja przełącznika⟩ ::= SWITCH ⟨identyfikator przełącznika⟩ :=⟨lista przełączeń⟩

5.3.2. Przykłady.

SWITCH S:= S1,S2,Q[M], IF V GREATER - 5 THEN S3 ELSE S4

SWITCH Q:= P1, W

5.3.3. Znaczenie.

Deklaracja przełącznika określa zbiór wartości przełączeń. Są to wartości kolejnych wyrażeń sterujących, wymienionych na liście przełączeń. Każdemu z tych wyrażeń odpowiada liczba naturalna 1,2,... będąca numerem porządkowym wyrażenia na liście przełączeń, przy czym wyrażenia liczone od lewej do prawej. Wartością przełączenia, odpowiadającą danej wartości wyrażenia indeksowego /punkt 3.5. "Wyrażenia sterujące "/, jest wartość tego wyrażenia sterującego, które ma na liście przełączeń numer równy wartości wyrażenia indeksowego.

5.3.4. Obliczanie wyrażeń na liście przełączeń.

Wyrażenie z listy przełączeń oblicza się za każdym razem, gdy program odwołuje się do elementu na liście, którym jest to wyrażenie. Przy obliczaniu tego wyrażenia korzysta się z aktualnych wartości wszystkich występujących w nim zmiennych.

5.3.5. Wpływ obszaru działania.

Jeżeli przełączenie występuje poza obszarem działania wielkości spotykanej w wyrażeniu sterującym na liście przełączeń i obliczenie tego przełączenia wybiera to wyrażenie sterujące, to kolizje między identyfikatorami wielkości z tego wyrażenia, a identyfikatorami, których deklaracje obowiązują w miejscu zajmowanym przez to przełączenie, usuwane są przez odpowiednią zmianę tych ostatnich identyfikatorów.

5.4. Deklaracje procedur.

5.4.1. Składnia.

⟨parametr formalny⟩ ::= ⟨identyfikator⟩

⟨lista parametrów formalnych⟩ ::= ⟨parametr formalny⟩|⟨lista parametrów formalnych⟩⟨ogranicznik parametru⟩⟨parametr formalny⟩

⟨zbiór parametrów formalnych⟩ ::=⟨puste⟩|(⟨lista parametrów formalnych⟩)

⟨lista identyfikatorów⟩::=⟨identyfikator⟩|⟨lista identyfikatorów⟩,

⟨identyfikator⟩

⟨lista parametrów wywoływanych przez wartość⟩ ::= VALUE ⟨lista identyfikatorów⟩ ;|⟨puste⟩

⟨specyfikacja⟩ ::= STRING|⟨typ⟩| ARRAY|⟨typ⟩ ARRAY| LABEL| SWITCH|
PROCEDURE|⟨typ⟩ PROCEDURE

⟨zbiór specyfikacji⟩ ::=⟨puste⟩|⟨specyfikacja⟩⟨lista identyfikatorów⟩ ;|
⟨zbiór specyfikacji⟩ ⟨specyfikacja⟩
⟨lista identyfikatorów⟩ ;

⟨nagłówek procedury⟩ ::= ⟨identyfikator procedury⟩⟨zbiór parametrów formalnych⟩ ;|⟨lista parametrów wywoływanych przez wartość⟩⟨zbiór specyfikacji⟩

⟨treść procedury⟩ ::= ⟨instrukcja⟩

⟨deklaracja procedury⟩ ::= PROCEDURE ⟨nagłówek procedury⟩ ⟨treść procedury⟩|⟨typ⟩ PROCEDURE ⟨nagłówek procedury⟩⟨treść procedury⟩

5.4.2. Przykłady.

```
PROCEDURE TRANSPOZYCJA (A) STOPIEN: (N) ; VALUE N; ARRAY A;
                                                INTEGER N;
    BEGIN REAL W; INTEGER I,K;
        FOR I:= 1 STEP 1 UNTIL N DO
            FOR K:= 1+I STEP 1 UNTIL N DO
```

```
            BEGIN W:= A[I,K]; A[I,K] := A[K,I];
                                        A[K,I] := W
            END
    END TRANSPOZYCJA

PROCEDURE ILOCZYNSKALARNY (A,B) STOPIEN:(K,P) WYNIK:(Y);
    VALUE K; INTEGER K,P; REAL Y,A,B;
    BEGIN REAL S;
      S:= 0; FOR P:=1 STEP 1 UNTIL K DO S:=S+A*B;
                                        Y:=S
    END ILOCZYNSKALARNY

PROCEDURE MAX (A) WYMIARY:(N,M) WYNIK:(Y) INDEKSY:(I,K);
    COMMENT ELEMENT MACIERZY A O WYMIARACH N NA M MA-
            JACY NAJWIEKSZA WARTOŚĆ BEZWZGLEDNA POSY-
            LA SIE DO Y. INDEKSY TEGO ELEMENTU UMIE-
            SZCZA SIE W ZMIENNYCH I,K;
    ARRAY A; INTEGER N,M,I,K; REAL Y;
    BEGIN INTEGER P,Q;
      Y:=0;
      FOR P:= 1 STEP 1 UNTIL N DO
       FOR Q:= 1 STEP 1 UNTIL M DO
        IF ABS (A[P,Q]) GREATER Y THEN
          BEGIN
            Y:= ABS (A[P,Q]); I:=P; K:=Q
          END
```

END MAX

5.4.3. Znaczenie.

Deklaracja procedury służy do określenia procedury związanej z identyfikatorem procedury. Zasadniczą częścią składową tej deklaracji jest instrukcja czyli treść procedury, do której można odwołać się z różnych części bloku, zawierającego na początku deklarację danej procedury, za pomocą odwołań funkcyjnych lub instrukcji procedury.
Z treścią procedury wiąże się nagłówek procedury, który określa pewne identyfikatory spotkane w treści procedury i reprezentujące parametry formalne. W momencie wywołania procedury /punkty 3.2. "Odwołanie funkcyjne" oraz 4.7. "Instrukcje procedury"/ w treści tej procedury parametrom formalnym przypisuje się wartości odpowiednich parametrów aktualnych albo zastępuje się parametry formalne odpowiednimi parametrami aktualnymi. Identyfikatory występujące w treści procedury, które nie są parametrami formalnymi, mogą być lokalne lub nielokalne w treści procedury - zależnie od tego czy są one tam zadeklarowane, czy nie.
Te z nich, które nie są lokalne w treści procedury, mogą być lokalne w bloku, na początku którego znajduje się deklaracja danej procedury. Treść procedury działa zawsze jak blok, nawet wtedy, gdy nie jest napisana w postaci bloku. Wobec tego obszar działania etykiety, którą opatrzono instrukcję w treści procedury lub samą treść, nie może wykraczać poza tę treść. Ponadto, jeżeli identyfikator

parametru formalnego zadeklarowano na nowo w treści procedury /włączamy tu przypadek użycia tego identyfikatora jako etykiety, jak w punkcie 4.1.3/, to identyfikator ten staje się przez to lokalny i odpowiadające mu parametry aktualne nie są dostępne w całym obszarze działania tej wewnętrznej lokalnej wielkości.

5.4.4. Wartości funkcji.

Na to, by deklaracja procedury określała wartość odwołania funkcyjnego, potrzeba by w treści procedury znajdowała się co najmniej jedna instrukcja przypisania, w której nazwa procedury występuje z lewej stony symbolu przypisania. Po wywołaniu procedury musi być wykonana co najmniej jedna taka instrukcja. Ta spośród wymienionych instrukcji, która zostanie wykonana ostatnia, określi wartośc odwołania funkcyjnego i ta wartość będzie używana do dalszego obliczania wyrażenia, w którym to odwołanie występuje. Typ identyfikatora procedury określa się przez użycie deklaratora typu REAL, INTEGER lub BOOLEAN jako pierwszego symbolu w deklaracji tej procedury. Każde wystąpienie identyfikatora procedury w jej treści, inaczej niż jako elementu listy lewych stron instrukcji przypisania, oznacza nowe odwołanie do tej procedury.

5.4.5. Specyfikacje.

Do nagłówka procedury należy włączyć zbiór specyfikacji, podający za pomocą oczywistych oznaczeń informacje o typach i klasach wszystkich parametrów formalnych. W tym zbiorze żaden parametr formalny nie może występować więcej niż raz.

6. STANDARDOWE PROCEDURY WEJŚCIA-WYJŚCIA.

6.1. Instrukcje wyjścia.

6.1.1. Składnia.

⟨instrukcja wyjścia⟩ ::= ⟨instrukcja wyjścia liczbowego⟩|
⟨instrukcja wyjścia logicznego⟩|
⟨instrukcja wyjścia tekstowego⟩|
⟨instrukcja wyjścia znakowego⟩|
⟨instrukcja rozmieszczenia⟩

⟨instrukcja wyjścia liczbowego⟩ ::= OUT(⟨numer symbolicznej operacji wyjścia⟩ ⟨ogranicznik parametru⟩ ⟨format liczbowy⟩ ⟨ogranicznik parametru⟩ ⟨lista liczbowych elementów wyjścia⟩)

⟨lista liczbowych elementów wyjścia⟩ ::= ⟨liczbowy element wyjścia⟩| ⟨lista liczbowych elementów wyjścia⟩ ⟨ogranicznik parametru⟩ ⟨liczbowy element wyjścia⟩

⟨liczbowy element wyjścia⟩ ::= ⟨wyrażenie arytmetyczne⟩| ⟨identyfikator macierzy⟩

⟨instrukcja wyjścia logicznego⟩::= OUT(⟨numer symbolicznej operacji wyjścia⟩ ⟨ogranicznik parametru⟩ ⟨format logiczny⟩ ⟨ogranicznik parametru⟩ ⟨lista logicznych elementów wyjścia⟩)

⟨lista logicznych elementów wyjścia⟩ ::= ⟨logiczny element wyjścia⟩| ⟨lista logicznych elementów wyjścia⟩ ⟨ogranicznik parametru⟩ ⟨logiczny element wyjścia⟩

⟨logiczny element wyjścia⟩ ::=⟨wyrażenie boolowskie⟩| ⟨identyfikator macierzy⟩

⟨instrukcja wyjścia tekstowego⟩::= OUT(⟨numer symbolicznej operacji wyjścia⟩ ⟨ogranicznik parametru⟩ 'T' ⟨ogranicznik parametru⟩ ⟨lista tekstowych elementów wyjścia⟩)

⟨lista tekstowych elementów wyjścia⟩::= ⟨tekstowy element wyjścia⟩| ⟨lista tekstowych elementów wyjścia⟩

⟨ogranicznik parametru⟩
⟨tekstowy element wyjścia⟩

⟨tekstowy element wyjścia⟩ ::= ⟨tekst⟩

⟨instrukcja wyjścia znakowego⟩ ::= OUTCHAR(⟨numer symbolicznej operacji wyjścia⟩ ⟨ogranicznik parametru⟩ ⟨znakowy element wyjścia⟩)

⟨znakowy element wyjścia⟩ ::= ⟨wyrażenie arytmetyczne⟩

⟨instrukcja rozmieszczenia⟩ ::= OUT(⟨numer symbolicznej operacji wyjścia⟩ ⟨ogranicznik parametru⟩ ⟨format rozmieszczenia⟩)| OUT (⟨ numer symboliczej operacji wyjścia⟩ ⟨ogranicznik parametru⟩ ⟨format rozmieszczenia⟩ ⟨ogranicznik parametru⟩ ⟨replikator rozmieszczenia⟩)

⟨replikator rozmieszczenia⟩ ::=⟨wyrażenie arytmetyczne⟩

⟨numer symbolicznej operacji wyjścia⟩ ::=⟨wyrażenie arytmetyczne⟩

6.1.2. Przykłady.

Instrukcje wyjścia liczbowego:

```
OUT (0, 'Z3DB3D5B', U,V,X [1], ENTIER (W))
OUT (P) FORMAT WYKŁADNICZY: ('E-.3DB3DE-2D'
        )PISZ WARTOŚĆ WYRAŻEŃ:(X POWER Y,
         IF BOOL THEN R [I,5] ELSE R [I+1,5])
```

```
OUT (1) STANDARDOWY FORMAT CAŁKOWITY: ('I' I, SILNIA (I))
```

Instrukcje wyjścia logicznego:

```
OUT (Q, '5F6B', B NOTGREATER D, TRUE,
     NOT (X LESS 2 AND Z GREATER 6 OR 2+Y=0))
OUT (0, '2BF2B', BOOL [J], BOOL [J+1])
```

Instrukcje wyjścia tekstowego:

```
OUT (5, 'T', ':5/:7BUKLAD:B ROWNAN: B NIEOZNACZONY')
OUT (0, 'T', ':7/BIURO:2BP:BR:BO:BJ:BE:BK:BT:B
          O:BW:/PRZEMYSLU:2BPAPIERNICZEGO:/',
          'LODZ,:2BUL.:B ZACHODNIA:2B 70 ')
```

Instrukcja wyjścia znakowego:

```
OUTCHAR (0) PISZ WARTOŚĆ KRESKI: (72)
OUTCHAR (N,M)
```

Instrukcja rozmieszczenia:

```
OUT (0, '3/2B')
OUT (0, "B', 10)
OUT (2, '/ ', LICZBA - 1)
```

6.1.3. Znaczenie.

Procedury wyjścia powodują wyprowadzenie na wyjście danych liczbowych, logicznych, tekstowych lub pojedyńczych symboli.

Pierwszy parametr instrukcji procedury wyjścia instrukcji wyjścia określa numer symbolicznej operacji wyjścia, który ma być wykorzystywany w trakcie wykonywania procedury. Sposób przypisania numerów symbolicznych operacji

```
OUT (1) STANDARDOWY FORMAT CAŁKOWITY: ('I; I, SILNIA (I))
```

Instrukcje wyjścia logicznego:

```
OUT (Q, '5F6B', B NOTGREATER D, TRUE,
     NOT (X LESS 2 AND Z GREATER 6 OR 2+Y=0))
OUT (0, '2BF2B', BOOL [J], BOOL [J+1])
```

Instrukcje wyjścia tekstowego:

```
OUT (5, 'T', ':5/:7BUKLAD:B ROWNAN: B NIEOZNACZONY')
OUT (0, 'T', ':7/BIURO:2BP:BR:BO:BJ:BE:BK:BT:B
              O:BW:/PRZEMYSLU:2BPAPIERNICZEGO:/',
              'LODZ,:2BUL.:B ZACHODNIA:2B 70 ')
```

Instrukcja wyjścia znakowego:

```
OUTCHAR (0) PISZ WARTOŚĆ KRESKI: (72)
OUTCHAR (N,M)
```

Instrukcja rozmieszczenia:

```
OUT (0, '3/2B')
OUT (0, "B', 10)
OUT (2, '/ ', LICZBA - 1)
```

6.1.3. Znaczenie.

Procedury wyjścia powodują wyprowadzenie na wyjście danych liczbowych, logicznych, tekstowych lub pojedyńczych symboli.

Pierwszy parametr instrukcji procedury wyjścia instrukcji wyjścia określa numer symbolicznej operacji wyjścia, który ma być wykorzystywany w trakcie wykonywania procedury. Sposób przypisania numerów symbolicznych operacji

wyjścia należy podać w czołówce JOM /patrz maszyna ZAM-41, Oprogramowanie System 141, Tom I/.

Drugi parametr instrukcji procedury wyjścia liczbowego, logicznego czy też tekstowego zadaje format wyprowadzania /punkty 6.1.4.1. oraz 6.1.5.1/, a następne parametry określają wyprowadzane elementy.

Liczbowym albo logicznym elementem wyjścia może być identyfikator macierzy. Wyprowadzenie elementów tej macierzy jest równoważne wykonaniu poniższego programu.
Przyjmijmy, że do macierzy C odnosiła się deklaracja:

```
ARRAY C [L0:U0,L1:U1,L2:U2,...,LK:UK]
```

wówczas realizowany jest cykl

```
FOR IK:=LK STEP 1 UNTIIL UK DO
  ...
   ...
    ...
     FOR I2:=L2 STEP 1 UNTIL U2 DO
      FOR I1:=L1 STEP 1 UNTIL U1 DO
       FOR I0:=L0 STEP 1 UNTIIL U0 DO
        OUT (N, ⟨format liczbowy⟩ , C [I0,I1,I2,...,IK])
```

gdzie N jest numerem symbolicznej operacji wyjścia. Jeżeli deklarowana macierz C jest typu BOOLEAN, to w instrukcji OUT wystąpić musi odpowiedni format logiczny.

6.1.4. Instrukcja wyjścia liczbowego.

6.1.4.1. Składnia formatu liczbowego.

⟨replikator⟩ ::= ⟨całkowita bez znaku⟩|⟨puste⟩

⟨składowa B⟩ ::= ⟨replikator⟩B|⟨składowa B⟩
⟨replikator⟩ B|⟨puste⟩

⟨składowa D⟩ ::= ⟨replikator⟩ D|⟨składowa D⟩⟨replikator⟩ D|⟨składowa D⟩⟨składowa B⟩|⟨puste⟩

⟨składowa Z⟩ ::= +|-|⟨puste⟩

⟨format całkowity⟩::=⟨składowa B⟩⟨składowa Z⟩⟨składowa D⟩

⟨format ułamkowy⟩ ::= ⟨format całkowity⟩|⟨format całkowity⟩.⟨składowa D⟩

⟨format wykładniczy⟩::=⟨format ułamkowy⟩ E ⟨format całkowity⟩

⟨format liczbowy⟩ ::= 'E'|'Y'|'Z'|'I'|'E⟨format wykładniczy⟩'|'Y⟨format ułamkowy⟩'|'Z⟨format ułamkowy⟩'

6.1.4.2. Przykłady.

```
'E'
'E2B-.3DB3DB3DB3DE+2D5B'
'E2D.4DE2D'
'E3B+3DBE-2D'
'Y'
'Y-3D.B4DB3D'
'YBDBDDDB3D'
'Z'
'Z-3D.B4DB3D'
'ZBDBDDDB3D'
'I'
```

6.1.4.3. Znaczenie formatu liczbowego.

Format liczbowy określa postać, w jakiej mają być wyprowadzane liczby dziesiętne będące wartościami poszczególnych elementów z listy elementów wyjścia liczbowego. Symbole używane do tworzenia formatów: całkowitego, ułamkowego i wykładniczego mają następujące znaczenie:

Symbol:	Znaczenie:
D	- wypisanie cyfry dziesiętnej
B	- wypisanie spacji
.	- wypisanie kropki pozycyjnej
E	- wypisanie litery E oddzielającej mantysę liczby od jej wykładnika
+	- wypisanie znaku liczby
-	- wypisanie znaku liczby, tylko dla liczb ujemnych

Konstrukcja "⟨replikator⟩B" (czy też "⟨replikator⟩ D") jest równoważna ciągowi składającemu się z takiej liczby liter B (lub D) jaką wskazuje replikator. Na przykład zapis 5B jest równoważny ciągowi BBBBB, a zapis 3D - równoważny ciągowi DDD.

Pierwsze litery formatu liczbowego mają następujące znaczenie:

Litera:	Znaczenie:
E	- wyprowadzenie liczby w formie wykładniczej, z pierwszą cyfrą różną od zera;

Y - wyprowadzenie wszystkich cyfr liczby zgodnie z formatem, tj, łącznie z zerami przed pierwszą cyfrą znaczącą;

Z - wyprowadzenie liczby z zastąpieniem jej początkowych zer spacjami i jednoczesnym przesunięciem znaku przed pierwszą cyfrą znaczącą.

Ponadto ustalone są standardowe formaty liczbowe, składające się z jednej litery, które mają następujące znaczenie:

Format standardowy:	Format równoważny:
'E'	'É3B+D.3DB3DB3DE+2D3B'
'Y'	'Y3B-6D.6D'
'Z'	'Ź3B-6D.6D'
'I'	'Y3B-B6D'

6.1.4.4. Ograniczenia.

Długość formatu liczbowego nie może przekraczać 48 symboli. Przez długość formatu rozumie się liczbę symboli wchodzących w skład formatu równoważnego, w którym nie występują replikatory. Ze względu na dokładność zmiennoprzecinkowych operacji arytmetycznych, liczba cyfr wypisywanej mantysy, w przypadku liczb zmiennoprzecinkowych lub liczba cyfr wypisywanej liczby, w pozostałych przypadkach, nie powinna przekraczać 12.

6.1.5. Instrukcja wyjścia logicznego.

6.1.5.1. Składnia formatu logicznego.

⟨składowa F⟩ ::= 5F|F

⟨format logiczny⟩ ::= 'L⟨składowa B⟩⟨składowa F⟩⟨składowa B⟩'

6.1.5.2. Przykłady.

'L3B5FB'

'LBF2B'

'L5F4B'

'LF2B'

'L5F'

'LF'

6.1.5.3. Znaczenie formatu logicznego.

Format logiczny określa szablon, według którego wyprowadzone są wartości logiczne, będące wartościami poszczególnych elementów listy elementów wyjścia logicznego.

Symbol:

5F - oznacza wypisanie wartości logicznej w postaci słowa FALSE albo TRUE, przy czym za słowem TRUE występuje dodatkowo jedna spacja,

F - oznacza wypisanie wartości logicznej w postaci litery F gdy wyprowadzana wielkość ma wartość FALSE albo T gdy wyprowadzana wielkość ma wartość TRUE .

6.1.6. Instrukcja wyjścia tekstowego.

Procedura wyjścia tekstowego powoduje wyprowadzenie tekstowych elementów wyjścia. Przy wypisywaniu pomija symbo-

le "'" ograniczające poszczególne teksty.

W tekstach występujących na liście tekstowych elementów wyjścia symbol ":" jest używany jedynie dla zmiany sensu symbolu występującego za replikatorem /znaczenie replikatora jest omówione w punkcie 6.1.4.3./.

Odbywa się to zgodnie z następującą tabelką:

Ciąg symboli:	Znaczenie na wyjściu:
: ⟨replikator⟩ B	- wypisanie spacji
: ⟨replikator⟩ /	- przejście do początku następnej linii
: ⟨replikator⟩ *	- przejście do początku następnej strony
: ⟨replikator⟩ O	- wypisanie symbolu "'"
: ⟨replikator⟩ Z	- wypisanie symbolu "'"
: ⟨replikator⟩ D	- wypisanie symbolu ":"

6.1.7. Instrukcja wyjścia znakowego.

Procedura wyjścia znakowego powoduje wyprowadzenie symbolu, którego kod określony jest wartością wyrażenia arytmetycznego, będącego znakowym elementem wyjścia /patrz dodatek B/.

6.1.8. Instrukcja rozmieszczenia.

6.1.8.1. Składnia formatu rozmieszczenia.

⟨znak rozmieszczenia⟩ ::= B| / |*

⟨wskaźnik rozmieszczenia⟩ ::=⟨replikator⟩⟨znak rozmieszczenia⟩|
⟨wskaźnik rozmieszczenia⟩
⟨replikator⟩
⟨znak rozmieszczenia⟩

⟨format rozmieszczenia⟩ ::= ⟨wskaźnik rozmieszczenia⟩

6.1.8.2. Przykłady.

'5/15BB'

'//7B'

'2/7B'

6.1.8.3. Znaczenie formatu rozmieszczenia.

Format rozmieszczenia określa sposób rozmieszczenia wyników na arkuszu wydawniczym.

Znaczenia symboli wykorzystywanych przy konstrukcji formatu rozmieszczenia są następujące:

znak B - oznacza spację;

znak / - oznacza przejście do początku następnego wiersza,

znak * - oznacza przejście do początku nowej strony.

Trzeci parametr procedury rozmieszczenia, jeśli występuje, określa liczbę powtórzeń formatu rozmieszczenia.

6.1.9. Arkusz wydawniczy.

Przy wydawnictwie wyników programista może się posługiwać standardowym albo niestandardowym arkuszem wydawniczym. Standardowy arkusz wydawniczy dla procedur wyjścia ALGOLu jest arkuszem dalekopisu lub arkuszem drukarki o następujących wymiarach:

liczba znaków w wierszu 68,

liczba wierszy na stronie 70,

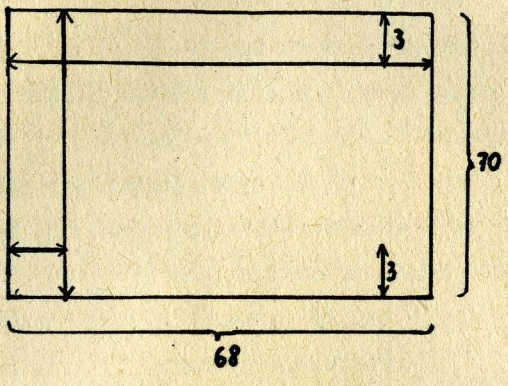
przy czym marginesy wynoszą:

lewy margines 4 znaki,

prawy margines 0 znaków,

górny margines 3 wiersze,

dolny margines 3 wiersze.



Drukowanie wyników rozpoczyna się od lewego marginesu i w przypadku przekroczenia przez wyprowadzoną wielkość prawego marginesu, cała jednostka jest przenoszona automatycznie do następnego wiersza, względnie na nową stronę.

Strona rozpoczyna się ciągiem 69 znaków "-" wypisanych w pierwszym wierszu, a w następnym wierszu umieszczany jest kolejny numer strony.

Pierwsze wykonanie procedury OUT albo OUTCHAR w danym programie powoduje wypisanie numeru strony równego 1. Rozmieszczenie poszczególnych wyników na arkuszu wydawniczym określa się za pomocą procedury rozmieszczenia /punkt 6.1.8/.

Wymiary niestandardowego arkusza wydawniczego można określić w czołówce języka operacyjnego ALGOLu /patrz punkt 7.1./.

6.2. Instrukcje wejścia.

6.2.1. Składnia.

⟨instrukcja wejścia⟩ ::= ⟨instrukcja wejścia danych⟩|
⟨instrukcja wejścia znakowego⟩

⟨instrukcja wejścia danych⟩ ::= INP(⟨numer symbolicznej
operacji wejścia⟩
⟨ogranicznik parametru⟩
⟨lista elementów wejścia⟩)

⟨lista elementów wejścia⟩ ::= ⟨element wejścia⟩|
⟨lista elementów wejścia⟩
⟨ogranicznik parametru⟩
⟨element wejścia⟩

⟨element wejścia⟩ ::= ⟨zmienna prosta⟩|⟨zmienna indeksowana⟩|⟨identyfikator macierzy⟩

⟨instrukcja wejścia znakowego⟩ ::= INCHAR(⟨numer symbolicznej operacji wejścia⟩)

⟨numer symbolicznej operacji wejścia⟩ ::= ⟨wyrażenie
arytmetyczne⟩

6.2.2. Znaczenie.

Procedury wejścia powodują wprowadzenie z wejścia danych liczbowych, logicznych lub pojedyńczych znaków kodu. Pierwszy parametr instrukcji procedury wejścia /instrukcji wejścia/ określa numer symbolicznej operacji wejścia. Sposób przypisania numerów symbolicznych operacji wejścia należy opisać w czołówce JOM /patrz ZAM-41, Oprogramowanie Systemu 141 Tom I/.

6.2.3. Instrukcja wejścia danych.

Procedura wejścia danych powoduje wprowadzenie wartości liczbowych lub logicznych i przypisanie tych wartości identyfikatorom, które występują jako dalsze parametry instrukcji. Każdemu ekementowi wejścia odpowiada jedna grupa danych liczbowych bądź logicznych, przy czym typ tych danych musi być, zgodny z typem elementu wejścia.

Jeżeli elementem wejścia jest identyfikator macierzy, to wprowadzanie danych liczbowych określających elementy tej macierzy jest równoważne wykonaniu poniższego programu.

Przyjmijmy, że do macierzy C odnosi się deklaracja:

```
ARRAY C[ L0:U0, L1:U1, L2:U2,...,LK:UK]
```

Wówczas realizowany jest cykl

```
FOR IK:=LK STEP 1 UNTIL UK DO
 ...
  ...
   ...
    FOR I2:=L2 STEP 1 UNTIL U2 DO
     FOR I1:=L1 STEP 1 UNTIL U1 DO
      FOR I0:=L0 STEP 1 UNTIL U0 DO
        INP(N,C [I0,I1,I2,...,IK])
```

gdzie N jest numerem symbolicznej operacji wejścia.

6.2.3.1. Składnia grupy danych.

⟨element danych liczbowych⟩ ::= ⟨liczba⟩|⟨komentarz⟩ ⟨liczba⟩

⟨grupa danych liczbowych⟩ ::= ⟨element danych liczbowych⟩ |

⟨element danych liczbowych⟩,
⟨grupa danych liczbowych⟩

⟨element danych logicznych⟩ ::=⟨wartość logiczna⟩|
⟨komentarz⟩
⟨wartość logiczna⟩

⟨grupa danych logicznych⟩ ::=⟨element danych logicznych⟩;|
⟨element danych logicznych⟩,
⟨grupa danych logicznych⟩

⟨komentarz otwarty⟩ ::=⟨litera⟩|⟨komentarz otwarty⟩
⟨dowolny znak różny od ":" oraz "="⟩

⟨komentarz⟩ ::=⟨komentarz otwarty⟩ :|⟨komentarz otwarty⟩ =

6.2.4. Procedura wejścia znakowego.

Procedura wejścia znakowego jest funkcją typu INTEGER, która powoduje wprowadzenie jednego znaku z urządzenia wejściowego i nadanie mu wartości, zgodnej z tablicą kodu KW6. /Dodatek B/

7. URUCHAMIANIE PROGRAMÓW.

7.1. Język operacyjny ALGOLu.

Każdy program napisany w języku ALGOL musi być poprzedzony czołówką napisaną w języku operacyjnym ALGOLu. Czołówkę stanowi dowolny zestaw zdań /może być pusty/ wybrany spośród opisanych poniżej zdań języka operacyjnego, zakończony symbolem ".". Poszczególne zdania czołówki należy oddzielić symbolem ";". W szczególności, czołówka może zawierać jedynie symbol ".". Do języka operacyjnego ALGOL należą następujące zdania:

TEKST

KONTROLA

WYDAWNICTWO: <część parametrowa>

WYPROWADŹ PROGRAM WYNIKOWY

7.1.1. Zdanie TEKST.

Zdanie tekst oznacza żądanie wypisania tekstu programu źródłowego na arkuszu drukarki lub na taśmie dalekopisowej, w zależności od określonego w czołówce JOM /patrz ZAM 41, Oprogramowanie /System 141/, Tom I/ numeru symbolicznej operacji wyjścia. Wypisywany tekst programu źródłowego jest przedzielany co dziesięć wierszy napisem: LINIA< liczba całkowita> , gdzie liczba całkowita jest numerem wiersza programu, następującego po tym napisie. Wiersze programu źródłowego numerowane są od zera.

Liczba wierszy programu nie może przekraczać 4095.

7.1.2. Zdanie KONTROLA.

Zdanie KONTROLA jest żądaniem tłumaczenia fragmentu tekstu programu źródłowego znajdującego się między symbolem "*" następującym bezpośrednio po słowie COMMENT, a symbolem ";". Jeżeli w czołówce brak jest zdania KONTROLA, to translator traktuje cały tekst występujący między słowem COMMENT, a symbolem ";" jako komentarz.

7.1.3. Zdanie WYDAWNICTWO.

7.1.3.1. Składnia.

⟨część parametrowa⟩::=⟨składowa części parametrowej⟩|⟨część parametrowa⟩,⟨składowa części parametrowej⟩

⟨składowa części parametrowej⟩::=⟨parametr 1⟩ = (⟨parametr 2⟩ , ⟨parametr 3⟩)

⟨parametr 1⟩::=⟨numer symbolicznej operacji wyjścia⟩

⟨parametr 2⟩::=⟨liczba znaków w wierszu⟩

⟨parametr 3⟩::=⟨liczba wierszy na stronie⟩

⟨numer symbolicznej operacji wyjścia⟩::= 0|1|2|3|4|5|6|7

⟨liczba znaków w wierszu⟩::=⟨liczba całkowita bez znaku⟩

⟨liczba wierszy na stronie⟩::=⟨liczba całkowita bez znaku⟩

7.1.3.2. Przykład.

WYDAWNICTWO: O=(120, 72),

1= (65, 30)

7.1.3.3. Znaczenie.

Zdanie WYDAWNICTWO określa niestandardowy format arkusza wydawniczego dla tych numerów symbolicznej operacji wyjścia, z których będą w programie korzystać instrukcje procedur OUT i OUTCHAR. Jeżeli w czołówce brak jest deklaracji wydawnictwa, to program wynikowy będzie korzystał z formatu standardowego. Liczba znaków w wierszu oraz liczba wierszy na stronie zostały ograniczone do 256.

7.1.4. Zdanie WYPROWADZ PROGRAM WYNIKOWY.

Zdanie WYPROWADZ PROGRAM WYNIKOWY jest żądaniem wypisania programu wynikowego na urządzeniu wyjściowym, którego numer symbolicznej operacji wyjścia był zdeklarowany w czołówce JOM. Ponieważ wyjście standardowe dla translatora ma numer 0, należy w przypadku żądania wyprowadzenia programu wynikowego, deklarować w czołówce JOM wyjście symboliczne numer 1.

7.2. Sygnalizacja błędów.

W czasie tłumaczenia programu źródłowego, jak i w trakcie wykonywania programu wynikowego, sprawdzana jest poprawność przetwarzanego tekstu i interpretowanego programu. Po wykryciu błędu wyprowadzona jest informacja określająca typ tego błędu oraz jego położenie w programie źródłowym.

Informacja opisująca błąd składa się z dwóch części wypisywanych w jednym wierszu:

BLAD <numer> LINIA <numer>

Liczba wypisana za słowem "BLAD" określa rodzaj wykrytego błędu, a liczba wypisana za słowem "LINIA" jest numerem wiersza programu źródłowego, w którym ten błąd wystąpił. Liczone są wszystkie wiersze tekstu napisanego w języku ALGOL /bez czołówki/, przy czym pierwszemu wierszowi programu przypisuje się numer 0.

Błąd w tekście programu, nie powoduje przerwania tłumaczenia dalszego ciągu tekstu programu przez dany przebieg translatora. Jeśli natomiast w trakcie tłumaczenia programu źródłowego nastąpi przepełnienie magazynów list bądź stosu, to praca translatora zostanie przerwana.

7.2.1. Błędy wykrywane w pierwszym przebiegu.

W pierwszym przebiegu translacji standaryzowane są symbole użyte w tekście programu źródłowego. Na tym etapie bada się poprawność wystąpienia komentarza, poprawność sąsiedztwa niektórych ograniczników oraz poprawność zapisu liczby.

W razie wykrycia błędu, po przejrzeniu tekstu programu do końca - w celu znalezienia ewentualnych dalszych błędów, praca translatora zostaje zatrzymana i wypisywany jest tekst:

BYLY BLEDY W PROGRAMIE ZRODLOWYM

Błędy i przepełniania list wykrywane i sygnalizowane w trakcie pracy pierwszego przebiegu translatora podane są w Dodatku C.

7.2.2. Błędy wykrywane w drugim przebiegu.

W drugim przebiegu translacji materiał przygotowany przez pierwszy przebieg przetwarzany jest na program wynikowy. Jeżeli w trakcie tych czynności zostanie wykryty błąd, to wypisywany jest odpowiedni sygnał, po czym blok, w którym znaleziono ten błąd przeglądany jest bez wykonywania jakiejkolwiek kontroli poprawności. Natomiast bloki wewnętrzne względem tego bloku są tłumaczone normalnie, z kontrolą poprawności, z tym, że analiza ich tekstów przeprowadzana jest tak, aby nie wykrywać ciągu błędów wynikających z deklaracji identyfikatorów w pomijanym bloku. Po opracowaniu materiału dotyczącego bloku zawierającego błąd następuje przejście do pełnej analizy dalszego tekstu.

Po przetworzeniu całego tekstu programu, w przypadku, gdy translacja przebiegała poprawnie pod względem formalnym, przechodzi się do wykonania dalszych czynności określonych w czołówce Języka Operacyjnego ALGOLu albo w czołówce JOM; w przypadku wykrycia błędu, translacja zostaje zatrzymana, analogicznie jak w pierwszym przebiegu.

Błędy i przepełnienia list wykrywane i sygnalizowane w trakcie pracy drugiego przebiegu translatora podane są w Dodatku D.

7.2.3. Błędy wykrywane w czasie realizacji programu wynikowego.

Po utworzeniu programu wynikowego można przystąpić do jego wykonania. Ze względu na to, że przyjęta metoda tłuma-

czenia nie pozwala wykryć wszystkich błędów, na przykład takich, że klasa i typ parametru aktualnego nie są zgodne z klasą i typem odpowiedniego parametru formalnego, wiele kontroli poprawności przeprowadza się w czasie wykonywania programu wynikowego.

W przypadku wykrycia błędu, dalsza praca interpretatora staje się niemożliwa. W związku z tym sygnalizowany jest błąd, następnie wypisywany jest tekst PRZYCZYNA ZAKONCZENIA-3, a maszyna przechodzi do wykonania następnej instrukcji określonej w czołówce JOM.

Spis błędów i przepełnień sygnalizowanych w trakcie wykonywania programu wynikowego podany jest w Dodatku E.

Po zakończeniu wykonywania programu wynikowego, w przypadku jego poprawnego wykonania, wypisywany jest tekst: PRZYCZYNA ZAKONCZENIA-0.

DODATEK A

SŁOWA ZASTRZEŻONE I ICH ZNACZENIE W JĘZYKU POLSKIM

Słowo zastrzeżone	Znaczenie	Słowo zastrzeżone	Znaczenie
AND	i	LABEL	etykieta
ARRAY	macierz	LESS	mniejsze
BEGIN	początek	NOT	nie
BOOLEAN	boolowski	NOTEQUAL	nierówne
COMMENT	komentarz	NOTGREATER	niewiększe
DIV	dziel całkowicie	NOTLESS	niemniejsze
DO	wykonaj	OR	lub
ELSE	inaczej	OWN	własny
END	koniec	POWER	potęga
EQUAL	równe	PROCEDURE	procedura
EQUIV	równoważne	REAL	rzeczywisty
FALSE	fałsz	STEP	krok
FOR	dla	STRING	tekst
		SWITCH	przełącznik
GOTO	idź do	THEN	to
GREATER	większe	TRUE	prawda
IF	jeżeli	UNTIL	dopóki
IMPL	implikuje	VALUE	wartość
INTEGER	całkowity	WHILE	podczas

DODATEK B

Reprezentacja znaków w kodzie wewnętrznym

znak	liczba	znak	liczba	znak	liczba	znak	liczba
puste	64	e0	80	L	96	0	112
spacja	65	e1	81	M	97	1	113
.	66	e2	82	N	98	2	114
(	67	e3	83	O	99	3	115
+	68	ml	84	P	100	4	116
*	69	A	85	Q	101	5	117
)	70	B	86	R	102	6	118
;	71	C	87	S	103	7	119
-	72	D	88	T	104	8	120
/	73	E	89	U	105	9	121
,	74	F	90	V	106	nl	122
:	75	G	91	W	107	◊	123
'	76	R	92	X	108	r3	124
=	77	I	93	Y	109	r4	125
r1	78	J	94	Z	110	r5	126
r2	79	K	95	dl	111		127

Znaki oznaczone symbolami r1, r2, r3, r4, r5, e0, e1, e2, e3, ml, dl i nl zależą od tego, z którym kodem aktualnie współpracuje urządzenie.

Na przykład, dla Międzynarodowego Kodu Nr 2 symbol:

r1 oznacza [

r2 oznacza]

r4 oznacza ↑

r5 oznacza CR powrót karetki

n1 oznacza LF nowa linia

Szczegółowa odpowiedniość znaków i liczb dla różnych kodów znajduje się w opisie maszyny użytkowej System Operacyjny, Tom DTR 05.01/41, część I, Opis maszyny Użytkowej; Instytut Maszyn Matematycznych, Warszawa 1967.

DODATEK C

BŁĘDY I PRZEPEŁNIENIA SYGNALIZOWANE W PIERWSZYM PRZEBIEGU

Nr błędu	Opis błędu
200	- wystąpienie nieprzewidzianego symbolu przed pierwszym ogranicznikiem "BEGIN" w programie
201	- wystąpienie po symbolu ":" jednego z symboli: ":=", ")", "'", "COMMENT"
202	- wystąpienie po symbolu "BEGIN" lub ";" jednego z symboli: ":", "=", ")", ","
203	- użycie symbolu "COMMENT" w niewłaściwym kontekście
204	- użycie symbolu "COMMENT" po symbolu "END"
205	- wystąpienie po symbolu ")" liczby albo jednego z symboli "COMMENT" lub ","
206	- niepoprawny zapis liczby
207	- niepoprawny zapis ogranicznika na liście argumentów procedury
208	- brak zakończenia programu symbolem "*"

W trakcie działania pierwszego przebiegu mogą wystąpić następujące przepełnienia:

Nr przepełnienia		Znaczenie przepełnienia
301	-	przepełnienie listy identyfikatorów
302	-	przepełnienie magazynu identyfikatorów składających się z więcej niż czterech symboli
303	-	przepełnienie magazynu tekstu
304	-	przepełnienie licznika wierszy
305	-	przepełnienie magazynu wyjścia

DODATEK D

BŁĘDY I PRZEPEŁNIENIA SYGNALIZOWANE W DRUGIM PRZEBIEGU

Nr błędu	Opis błędu
100	- brak ogranicznika /albo ogranicznik błędnie wypisany/ między dwoma identyfikatorami, liczbami lub wartościami logicznymi albo też ogranicznik użyty w niewłaściwej strukturze. Przykład: `REAL PROCEDURE IMPL ( X,Y,Z) ;` `U := ALPHA BETA;`
101	- specyfikator użyty w niewłaściwym kontekście. Na przykład, słowo zastrzeżone LABEL użyte jako identyfikator
102	- błąd w kontekście zawierającym ogranicznik "BEGIN"
104	- deklarator występuje w niewłaściwym kontekście
105	- błąd w zbiorze parametrów formalnych; na przykład, powtórne wystąpienie identyfikatora jako parametru formalnego
106	- błąd w liście parametrów wywoływanych przez wartość albo zbiorze specyfikacji

Nr błędu	Opis błędu
107	- użycie w niewłaściwej strukturze operatora arytmetycznego, logicznego lub relacji bądź też jednego z ograniczników: "GOTO", "FOR", "IF", "(", ":". Przykład: FOR L=1 STEP 1 UNTIL N DO A[L]:=1;
108	- ogranicznik "END" jest użyty jako symbol zamykający deklarację. Przykład: BEGIN REAL X,Y,Z END
109	- błąd w konstrukcji poprzedzającej ogranicznik "END"
110	- brak średnika kończącego ostatnią deklarację
111	- błąd powodujący niewłaściwą konstrukcję bloku
112	- niewłaściwie użyty operator relacji lub zmienna indeksowana. Przykład: X LESS Y+Z LESS 3
114	- w kontekście zamiast wywołania procedury bez parametrów znajduje się coś innego
115	- ";" kończy nieprawidłowo zbudowaną strukturę
116	- błąd w deklaracji przełącznika
117	- błąd w deklaracji macierzy

Nr błędu	Opis błędu
118*	- niezgodność wystąpienia identyfikatora z jego deklaracją
119*	- niezgodność wystąpienia identyfikatora. Przykład: IF U NOTEQUAL U [I]
120	- błędny kontekst zawierający ":="
121	- niepoprawne wyrażenie występujące w deklaracji macierzy
122	- na liście przełączeń niepoprawnie wypisane wyrażenie sterujące
123	- operator arytmetyczny nie jest poprzedzony właściwym symbolem. Przykłady: I:=*R; OUT(0,'E', A +-12.5) ;
124	- niepoprawny kontekst zawierający separator ":"
125	- wystąpienie operatora relacji w wyrażeniu arytmetycznym
126	- niepoprawna konstrukcja wyrażenia zawierającego warunek

* przy błędach oznaczonych * wypisywane są w pewnych przypadkach numery dwóch wierszy; 1-szy z nich wskazuje koniec aktualnie przetwarzanego bloku, a 2-gi wiersz, w którym występuje źle użyty identyfikator.

Nr błędu	Opis błędu
127	- ogranicznik "THEN" nie jest poprzedzony przez wyrażenie boolowskie albo wyrażenie to nie jest poprawnie zapisane
128	- niepoprawna budowa instrukcji warunkowej albo wyrażenia warunkowego
129	- ogranicznik "GOTO" występuje w nieprawidłowym kontekście
130	- operator logiczny występuje w wyrażeniu arytmetycznym albo niepoprawna budowa wyrażenia boolowskiego
131	- niekompletna struktura poprzedzająca ogranicznik "FOR". Przykłady: X:=X+1 FOR I:=0 STEP 1 UNTIL X DO A[I] :=0; FOR I:=1, 10 FOR J:=0 STEP 1 UNTIL 10 DO A [I,J] :=0;
132	- separator "STEP", "UNTIL" albo "WHILE" występuje w nieprawidłowym kontekście albo po separatorze "STEP" nie występuje "UNTIL"
133	- błąd w kontekście zawierającym "("; na przykład, brak operatora
134	- ")" występuje w niewłaściwym kontekście
136	- niepoprawnie zbudowane wyrażenie indeksowe

Nr błędu	Opis błędu
137	- niepoprawna konstrukcja par ograniczeń albo wyrażenia indeksowego
140	- niepoprawnie zbudowane wyrażenie na liście cyklu
141	- niewłaściwa budowa parametru aktualnego
142	- wywołanie w wyrażeniu procedury, która nie jest nazwą funkcji
143	- nawias "[" nie jest poprzedzony identyfikatorem lub zmienna indeksowana czy przełączenie występuje w błędnym kontekście
144	- w parach ograniczeń macierzy typu własnego użyty jest identyfikator albo wystąpuje niewłaściwy znak
145	- błędnie użyty ogranicznik w parach ograniczeń macierzy własnej
146	- za duża liczba elementów macierzy własnej
147	- separator ":" lub "," występuje w niepoprawnym kontekście
148	- błędnie użyty separator ":" lub "," w deklaracji macierzy
149	- separator ":" występuje w wyrażeniu indeksowym

Nr błędu	Opis błędu
*150	- liczba indeksów zmiennej indeksowanej nie odpowiada liczbie indeksów ustalonych w deklaracji macierzy albo liczba parametrów formalnych nie jest równa liczbie parametrów aktualnych
151	- brak identyfikatora na liście deklaracji zmiennych prostych. Przykład: REAL X, ,Y;
152	- wyrażenia dla ograniczeń indeksów zależą od zmiennej lokalnej bądź funkcji lokalnej w bloku, w którym obowiązuje deklaracja macierzy
153	- powtórna deklaracja tego samego identyfikatora
154	- niewłaściwa budowa bloku
*155	- użycie identyfikatora niezgodne z jego deklaracją
*156	- podstawienie na identyfikator procedury funkcji poza treścią tej procedury
*157	- podstawienie na identyfikator procedury nie będącej funkcją
*158	- użycie identyfikatora niezgodne z jego deklaracją

* patrz odnośnik na str.D-3

Nr błędu	Opis błędu
*159	- w niewłaściwy sposób wywołana jest procedura bez parametrów, nie będąca funkcją
*160	- wywołanie procedury niefunkcyjnej
*161	- niewłaściwe wywołanie bezparametrowej procedury
162	- brak specyfikacji jednego z parametrów formalnych
163	- na liście parametrów wywoływanych przez wartość występuje parametr nie mający wartości
164	- powtórne wyszczególnienie parametru formalnego w specyfikacjach
165	- powtórne wyszczególnienie parametru formalnego na liście parametrów wywoływanych przez wartość
166	- brak identyfikatora z lewej strony ogranicznika ":="
167	- wartość logiczna występuje w wyrażeniu arytmetycznym
168	- brak identyfikatora, liczby lub wartości logicznej między ogranicznikami w wyrażeniu

* patrz odnośnik na str. D-3

Nr błędu	Opis błędu
169	- wystąpienie nawiasów " (" ") " w błędnym kontekście
170	- wyrażenia dla ograniczeń indeksów zależą od zmiennej lokalnej bądź funkcji lokalnej w bloku, w którym obowiązuje deklaracja macierzy
171	- podstawienie na identyfikator procedury poza treścią tej procedury
172	- błędne wywołanie procedury
173	- błędne wywołanie procedury bez parametrów
174	- błąd w kontekście zawierającym tekst. Przykład: OUT(0, `T``MASZYNA:BZAM-41')
175	- niezadeklarowany identyfikator albo identyfikator występuje w specyfikacjach, a nie występuje jako parametr formalny
176	- powtórna deklaracja etykiety

W trakcie działania drugiego przebiegu wystąpić mogą następujące przepełnienia:

Nr przepełnienia		Znaczenie przepełnienia
306	-	przepełnienie stosu
307	-	brak miejsca na bębnie na program wynikowy
308	-	brak miejsca na bębnie na listę nazw
310	-	błąd maszyny lub błąd wynikający z poprzednich błędów
311	-	błąd maszyny lub błąd wynikający z poprzednich błędów
312	-	błąd maszyny
313	-	za duży poziom bloku

DODATEK E

BŁĘDY I PRZEPEŁNIENIA SYGNALIZOWANE W TRAKCIE WYKONYWANIA PROGRAMU WYNIKOWEGO

Nr błędu	Opis błędu
*1	- liczba parametrów aktualnych niezgodna z liczbą parametrów formalnych
2	- niewłaściwa wielkość na liście lewych stron w instrukcji przypisania lub w warunku cyklu
3	- wyrażenie występujące w instrukcji przypisania lub element listy cyklu nie jest odpowieniego typu
4	- niezgodność typów w instrukcji przypisania lub w warunku cyklu
5	- zmienne i identyfikatory procedury występujące na liście lewych stron w instrukcji przypisania nie mają wspólnego typu
7	- niewłaściwy typ zmiennej przed operatorem w wyrażeniu arytmetycznym lub w relacji
8	- niewłaściwy typ zmiennej po operatorze w wyrażeniu arytmetycznym lub w relacji

* pierwszy parametr za tekstem LINIA, określa wiersz, w którym rozpoczyna się deklaracja procedury, drugi parametr za tekstem LINIA określa miejsce gdzie wywołano tę procedurę.

Nr błędu	Opis błędu
9	- niewłaściwy typ zmiennej występującej po znaku "-" w wyrażeniu arytmetycznym
10	- dzielna nie jest typu arytmetycznego
11	- dzielnik nie jest typu arytmetycznego
12	- w warunku nie występuje wyrażenie boolowskie
13	- w elemencie "podczas gdy" za ogranicznikiem "WHILE" nie występuje wyrażenie boolowskie
14	- nieokreślony wynik potęgowania a^i /a = 0, i = 0/
15	- niewłaściwy typ podstawowy a
16	- niewłaściwy typ wykładnika i
17	- niekreślony wynik potęgowania: podstawa ujemna, wykładnik rzeczywisty
18	- niewłaściwy typ argumentu operacji "NOT"
*19	- niewłaściwe podstawienie na parametr formalny
*20	- niewłaściwe podstawienie na parametr formalny

* patrz odnośnik na stronie E-1

Nr błędu	Opis błędu
*21	- niewłaściwe podstawienie na parametr formalny /wielkość nie odpowiada specyfikacji REAL/
* 22	- niewłaściwe podstawienie na parametr formalny /wielkość nie odpowiada specyfikacji BOOLEAN/
* 23	- niewłaściwe podstawienie na parametr formalny /wielkość nie odpowiada specyfikacji LABEL/
*24	- niewłaściwe podstawienie na parametr formalny /wielkość nie odpowiada specyfikacji INTEGER lub REAL/
*25	- niewłaściwe podstawienie na parametr formalny, który w treści procedury występuje jako element listy lewych stron w instrukcji przypisania, a specyfikowany jest jako REAL
*26	- niewłaściwe podstawienie na parametr formalny, który w treści procedury występuje jako element listy lewych stron w instrukcji przypisania, a specyfikowany jest jako INTEGER

* patrz odnośnik na stronie E-1

Nr błędu	Opis błędu
*27	- niewłaściwe podstawienie na parametr formalny, umieszczony na liście parametrów wywoływanych przez wartość a specyfikowany jako REAL
*28	- niewłaściwe podstawienie na parametr formalny, umieszczony na liście parametrów wywoływanych przez wartość i specyfikowany jako INTEGER
*29	- niewłaściwe podstawienie na parametr formalny, umieszczony na liście parametrów wywoływanych przez wartość i specyfikowany jako BOOLEAN
*30	- niewłaściwe podstawienie na parametr formalny, umieszczony na liście parametrów wywoływanych przez wartość i specyfikowany jako LABEL
31	- lewy argument operatora DIV nie jest typu INTEGER
32	- prawy argument operatora DIV nie jest typu INTEGER
33	- lewy argument operatora logicznego AND, OR, IMPL, EQUIV nie jest typu BOOLEAN

* patrz odnośnik na stronie E-1

Nr błędu	Opis błędu
34	- prawy argument operatora logicznego nie jest typu BOOLEAN
35	- ograniczenie dolne w deklaracji macierzy nie jest typu arytmetycznego
36	- ograniczenie górne w deklaracji macierzy nie jest typu arytmetycznego
37	- wartość ograniczenia górnego jest mniejsza od wartości ograniczenia dolnego
38	- wartość indeksu większa od 32767
39,40	- identyfikator określa niewłaściwą klasę wyrażenia indeksowego
41	- błędnie podane wartości indeksów, element macierzy wykracza poza zarezerwowany dla macierzy obszar
42	- liczba indeksów nie jest zgodna z deklaracją odpowiedniej macierzy
¤43	- niewłaściwe podstawienie na parametr formalny, umieszczony na liście parametrów wywoływanych przez wartość, a specyfikowany jako REAL albo INTEGER ARRAY
¤44	- niewłaściwe postawienie na parametr

¤ patrz odnośnik na stronie E-1

Nr błędu	Opis błędu
	formalny, umieszczony na liście parametrów wywoływanych przez wartość, a specyfikowany jako BOOLEAN ARRAY
45	- niewłaściwy parametr aktualny przy wywołaniu funkcji standardowej
46	- niewłaściwie określony parametr aktualny reprezentujący numer symbolicznej operacji wejścia-wyjścia w instrukcjach procedur INP, OUT lub OUTCHAR
47	- niewłaściwe podstawienie na parametr określający element wejścia-wyjścia
48	- parametr aktualny przy wywołaniu procedury OUT nie jest tekstem
49	- niewłaściwa liczba parametrów aktualnych przy wywołaniu funkcji standardowej
50	- błąd w wywołaniu procedury wyjścia OUT
51	- błąd w wywołaniu procedury wejścia INP
57	- dzielnik jest zerem
58	- przekroczenie zakresu liczb
61	- użycie procedury OUT INP i OUTCHAR jako parametru aktualnego
62	- użycie instrukcji procedury OUT w treści

Nr błędu	Opis błędu
	procedury obliczającej wartość odwołania funkcyjnego, które występuje w wyrażeniu arytmetycznym lub boolowskim - będącym elementem wyjścia
78	- argument, dla którego ma być wyliczona wartość funkcji SQRT jest niewiększy od 0
79	- za duży argument, dla którego ma być wyliczona wartość funkcji EXP
80	- argument, dla którego ma byc wyliczona wartość funkcji LN jest ujemny
87	- błąd maszyny
88	- na wejściu brak elementu wejścia, który powinien być wartością logiczną
89	- błędny znak przy czytaniu elementu logicznego
90	- błąd maszyny
91	- liczba za duża dla maszyny
92	- błędny znak przy czytaniu liczby
93	- za dużo znaków w formacie rozmieszczenia
95	- błędny znak w formacie liczbowym
96	- błędny znak na rozpoczynający lub kończą-

Nr błędu	Opis błędu
	- cy format
97	- błędny znak w replikatorze
98	
99	- za duży numer symbolicznej operacji wyjścia

Podczas wykonywania programu wynikowego mogą wystąpić następujące przepełnienia:

Nr przepełnienia	Znaczenie przepełnienia
314	- przekroczony obszar BEB
315	- program nie mieści się w PAO
316	- przepełnienie stosu
317	- przepełnienie stosu
319	- za duży numer segmentu
320	- błąd maszyny
321	- błąd maszyny
400	- błąd przesłania z BEB