ZAM-41

Język programowania

algol

INSTYTUT MASZYN MATEMATYCZNYCH BOITE

Język programowania
ALGOL
dla ZAM 41

OPIS

WYDAWNICTWA PRZEMYSŁU MASZYNOWEGO „WEMA"
Warszawa 1971

Autorzy:

mgr Jowita Koncewicz, mgr Maria Łącka

Adres Redakcji:

Warszawa, ul. Krzywickiego 34
tel. 21-84-41 w. 431 lub 28-37-29

WPM"WEMA". Warszawa 1971. Wydanie I.Nakład 1500+45 egz.
Ark.wyd.6,0.Ark.druk.8,0.Papier offset III kl. 80 g.
Format 100x197 mm. Zamówienie 413/71-5-2/S

WPM"WEMA"Zakład Poligraficzny w B-stoku.Zam.161. U-97

Spis rzeczy

1. WSTĘP

Opisany niżej język ALGOL jest reprezentacją konkretną języka ALGOL 60 dla maszyny cyfrowej ZAM 41. W rozdziałach 2-5 podano definicje jednostek składniowych języka ALGOL, reguły tworzenia struktur poprawnych w tym języku, objaśnienia znaczenia poszczególnych konstrukcji językowych oraz informacje o specyficznych ograniczeniach przyjętych w tej reprezentacji języka. Rozdziały te są tłumaczeniem odpowiednich rozdziałów oficjalnego dokumentu "Revised Report on the Algorithmic Language ALGOL 60"[1] zachowującym jego układ, lecz zawierającym wiele uzupełnień oraz uwzględniającym wszystkie zmiany wynikłe z przyjętej reprezentacji. W tłumaczeniu wzorowano się na tłumaczeniu powyższego dokumentu przez S. Paszkowskiego[2], starano się też zachować przyjętą w nim terminologię, wprowadzając zmiany tylko w niektórych przypadkach. Część z nich, to zmiany zatwierdzone przez

---

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

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

zespół C Komisji d/s Systemów Oprogramowania Maszyn Cyfrowych[1], pozostałe wynikły z konieczności zastosowania jednolitej terminologii w opisach wszystkich elementów oprogramowania maszyny cyfrowej ZAM 41.

Rozdział 6 zawiera opis standardowych procedur wejścia-wyjścia ALGOLu uzupełniony przykładami danych wejściowych i wyjściowych na oryginalnych formularzach. W rozdziale 7 podano wiadomości niezbędne do uruchamiania programów w ALGOLu dla maszyny cyfrowej ZAM 41, a w rozdziale 8 - bardziej szczegółowe informacje na temat Translatora ALGOLu i pracy w Systemie Oprogramowania 141 - tzw. SO 141.

Rozdział 9 jest przeznaczony dla tych Czytelników, którzy chcieliby poznać jedynie różnice między językiem wzorcowym, a opisywaną reprezentacją konkretną ALGOLu. Wymieniono w nim wszystkie zmiany w porządku numeracji punktów opisu języka wzorcowego.

W rozdziale 10 zamieszczono kilka przykładów programowania problemów w języku ALGOL. Przy doborze przykładów kierowano się przede wszystkim możliwościami przejrzystej ilustracji poszczególnych zagadnień związanych z użytkowaniem opisywanego języka, a nie ich wartością numeryczną.

1.1. Formalizm opisu składni

Składnia języka (tj. obiekty określone w języku) została opisana w notacji zaproponowa-

---

[1] Protokół z posiedzenia zespołu C Komisji d/s Systemów Oprogramowania Maszyn Cyfrowych odbytego dn. 8 maja 1968 w Zakopanem podczas Konferencji GAMS GAJAPEI.

nej przez J.W. Backusa[1] - za pomocą formuł metajęzykowych. Formuły metajęzykowe składają się ze zmiennych metajęzykowych, stałych metajęzykowych i łączników metajęzykowych.

Łączniki metajęzykowe oznacza się symbolami ::= oraz | i czyta odpowiednio jako "równe z definicji" i "lub". Symbol ::= dzieli formułę na dwie części. Z jego lewej strony umieszcza się obiekt definiowany, z prawej - 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 same siebie, 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 umieszcza się w nawiasach kątowych, nie umieszcza się w nich natomiast symboli podstawowych.

Na przykład formuła

<ab> ::= /|:|<ab> /|<ab><d>

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

/

lub

:

---

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

lub

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

lub

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

Niech wartościami zmiennej < d > będą cyfry dziesiętne

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

wówczas niektórymi wartościami zmiennej metajęzykowej < ab > będą:

:/
///
:///1
:///1/
:///1/37
/12345
:86

Definicja:

< puste > ::=

oznacza pusty ciąg symboli.

1.2. Translator języka ALGOL

Translator języka ALGOL opracowany dla maszyny cyfrowej ZAM 41 jest adaptacją translatora Whetstone Compiler dla maszyny cyfrowej KDF 9[1)], zrealizowaną z wieloma modyfikacjami wynikają-

1) Randell, B. and Russell, L.J., ALGOL 60 Implementation, AP, London and New York 1964.

cymi 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 za pomocą programu zwanego Interpreterem. Rozmieszczenie danych w pamięci operacyjnej dokonuje się dynamicznie za pomocą programowanego stosu podczas wykonywania programu wynikowego. Zarówno program wynikowy, jak i stos są automatycznie segmentowane i umieszczane w pamięci bębnowej, skąd Interpreter sprowadza je do pamięci operacyjnej.

Translator ALGOLu dla maszyny cyfrowej ZAM 41 opracował zespół pod kierownictwem Marii Łąckiej w składzie: Stanisław Choromański, Jowita Koncewicz, Janina Kozłowska-Pawlikowska, Waldemar Romaniuk i Zbigniew Zorski.

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

Programy w języku ALGOL tworzy się 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).

2.3. Ograniczniki

< ogranicznik > ::= < operator > | < separator > | < nawias > | < deklarator > | < specyfikator >

< operator > ::= < operator arytmetyczny > | < 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 | ARRAY | SWITCH | PROCEDURE

< specyfikator > ::= STRING | LABEL | VALUE

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

2.3.1. Słowa zastrzeżone

Ograniczniki będące słowami utworzonymi z liter, a także wartości logiczne, są słowami zastrzeżonymi, tzn. nie można ich używać jako identyfikatorów[1]. Wewnątrz słowa zastrzeżonego nie

[1] Zastrzeżenie to nie dotyczy separatora E i poszczególnych składowych operatora GO TO - można używać identyfikatorów: GO, TO i E.

może występować spacja ani nowa linia. Między słowami GO i TO może pojawić się dokładnie jedna spacja, ale nie może występować nowa linia. Jeżeli bezpośrednio 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 znaczenie w języku polskim umieszczono w Dodatku A.

Przyjmuje się, że następujące ograniczniki są równoważne:

Ogranicznik	Jego równoważnik
GOTO	GO TO
EQUAL	=

W ograniczniku ":=" między symbolami ":" i "=" nie może występować ani spacja, ani nowa linia.

2.3.2. Komentarze

Następujące reguły pozwalają na wstawienie komentarza między symbole programu:

Ciąg symboli podstawowych	Jego równoważnik
; COMMENT < ciąg dowolnych symboli nie zawierający ; > ;	;
BEGIN COMMENT < ciąg dowolnych symboli nie zawierający ; > ;	BEGIN
END < ciąg dowolnych symboli nie zawierający ; ani END, ani ELSE >	END

Równoważność oznacza tu, że translator zastąpi każdą z trzech struktur[1] podanych w le-

[1] Nie dotyczy to struktur zawartych w tekstach - patrz p. 2.6. "Teksty".

wej kolumnie odpowiednim symbolem z prawej kolumny; zastąpienie to nie będzie miało jakiegokolwiek wpływu na program. Przyjmuje się też, że pierwszy komentarz (przy czytaniu symboli programu od lewej strony wiersza do prawej) ma pierwszeństwo w zastępowaniu go równoważnikiem przed strukturami znajdującymi się w dalszym ciągu programu. Szczególna postać komentarza, a mianowicie komentarz rozpoczynający się od słowa COMMENT, po którym następuje bezpośrednio symbol * , może służyć do warunkowego włączenia instrukcji lub deklaracji do programu (patrz punkty 7.1. "Język operacyjny ALGOLu" i 7.1.2. "Zdanie KONTROLA").

2.3.2.1. P r z y k ł a d

```
BEGIN COMMENT PROGRAM ILUSTRUJACY
      STOSOWANIE KOMENTARZY;
      INTEGER I; REAL X;
      COMMENT POCZATEK PROGRAMU BEGIN
      FOR I := 1 STEP 1 UNTIL 10 DO
      OBLICZENIA END KONIEC PROGRAMU;
      FOR I := 1 STEP 1 UNTIL 10 DO
          BEGIN  IF I = 5
              THEN BEGIN X := I * (I + 1);
                         X := X POWER X
                   END PRZYPADEK 1
              ELSE BEGIN X := I * (I - 1);
                         X := X POWER(X + 1)
                   END PRZYPADEK 2
          END
END KONIEC PROGRAMU *
```

Program powyższy jest równoważny następującemu:

```
BEGIN INTEGER I; REAL X;
      FOR I := 1 STEP 1 UNTIL 10 DO
          BEGIN IF I = 5
             THEN BEGIN X := I * (I + 1);
                        X := X POWER X
                  END
             ELSE BEGIN X := I * (I - 1);
                        X := X POWER (X + 1)
                  END
          END
END *
```

2.4. Identyfikatory

2.4.1. Składnia

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

2.4.2. Przykłady

Identyfikatory zbudowane poprawnie:

A
BETA
ALFA5
PIERWIASTKIWIELOMIANU
D12MP32F

Identyfikatory zbudowane niepoprawnie:

ALFA 5
1B
PIERWIASTKI WIELOMIANU

2.4.3. Znaczenie

Identyfikatory nie mają samoistnego sensu, lecz służą jedynie do nazywania zmiennych prostych, tablic, etykiet, przełączników i procedur. Można je wybierać dowolnie, jednak z uwzględnieniem ograniczeń wymienionych w punk-

tach 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

W jednym programie można deklarować najwyżej 511 różnych identyfikatorów. 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 całko-
    wita 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. Ograniczenia nałożone na liczby podano w punkcie 3.3.6 "Arytmetyka wielkości typu REAL i INTEGER".

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 nie zawierający '>'

2.6.2. Przykłady

```
'I'
'+K,-] =+(⌈17↑) 125'
'PIERWIASTKI: 2BWIELOMIANU'
```

2.6.3. Znaczenie

Teksty stosuje się jako parametry standardowych procedur wyjścia (punkt 6.1. "Instrukcje wyjścia").

Tekst może zawierać najwyżej 384 symbole (wraz z obydwoma symbolami "'").

Symbole spacji i nowej linii są pomijane podczas czytania tekstu, nie mają one bowiem żadnego znaczenia w obrębie tekstu.

2.7. Wielkości, klasy i obszary działania

Rozróżnia się następujące klasy wielkości: zmienne proste, tablice, etykiety, przełączniki i procedury.

Obszarem działania wielkości jest zbiór instrukcji i wyrażeń, w którym obowiązuje deklaracja identyfikatora tej wielkości. Etykiety omówiono w punkcie 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 zdefiniowano w rozdziale 3.

Wartością identyfikatora tablicy jest uporządkowany zbiór wartości odpowiedniej tablicy 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 w języku ALGOL są wyrażenia arytmetyczne, boolowskie i sterujące. Elementami tych wyrażeń, oprócz odpowiednich ograniczników są wartości logiczne, liczby, zmienne, odwołania funkcyjne oraz operatory elementarne: arytmetyczne, relacji, logiczne i następstwa. Ponieważ w definicjach składni zmiennej i odwołania funkcyjnego występuje wyrażenie, 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 tablicy⟩ ::= ⟨identyfikator⟩
⟨zmienna indeksowana⟩ ::= ⟨identyfikator tablicy⟩[⟨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 pojedynczej wartości. Z wartości tej można korzystać w wyrażeniach w celu 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 tablicy (patrz punkt 5.2. "Deklaracje tablic").

3.1.4. Indeksy

3.1.4.1. Zmienne indeksowane są nazwami wartości elementów tablic (patrz punkt 5.2. "Deklaracje tablic"). 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 "[" i "]". Element tablicy, odpowiadający zmiennej indekso-

wanej, określają 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 tablicy (patrz punkt 5.2. "Deklaracje tablic").

3.1.4.3. O g r a n i c z e n i e

Wartość bezwzględna indeksu nie może przekraczać liczby 32767.

3.2. Odwołanie funkcyjne

3.2.1. Składnia

```
<identyfikator procedury> ::= <identyfikator>
<parametr aktualny> ::= <tekst>|<wyrażenie> |
    <identyfikator tablicy>|<identyfikator
    przełącznika>|<identyfikator procedury>
<tekst literowy> ::= <litera>|<tekst litero-
                     wy><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)
R
S(X-Y) TEMPERATURA: (T) CISNIENIE: (C)
INCHAR(O)
SIGN(B POWER 2 - 4 * A * C)
```

3.2.3. Znaczenie

Odwołanie funkcyjne określa pojedynczą 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 podano w punkcie 4.7. ("Instrukcje procedury"). Nie każda deklaracja procedury określa wartość odwołania funkcyjnego.

3.2.3.1. O g r a n i c z e n i e

W ograniczniku parametru postaci:

)< tekst literowy >:(

nie może występować symbol nowej linii.

3.2.4. Funkcje standartowe

Jeżeli w programie nie występują deklaracje (patrz rozdział 5. "Deklaracje") odnoszące się do wymienionych poniżej identyfikatorów, to wówczas te identyfikatory są nazwami funkcji standartowych ALGOLu.

ABS
ARCTAN
COS
ENTIER
EXP
INCHAR
LN
SIGN
SIN
SQRT

W przypadku, gdy programista nie chce korzystać w programie lub jego części z jakiejś funkcji standardowej, może wtedy używać jej nazwy jako identyfikatora zadeklarowanej w danym programie wielkości.

3.2.4.1. Znaczenie

Poniższe odwołania do funkcji standardowych oznaczają odpowiednio:

ABS(W) wartość bezwzględną wartości wyrażenia W;

ARCTAN(W) wartość główną funkcji argus tangens W;

COS(W) wartość funkcji cosinus W;

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

EXP(W) wartość funkcji wykładniczej wartości W;

INCHAR(W) wartość symbolu wprowadzanego z urządzenia wejściowego związanego z numerem symbolicznej operacji wejścia W;

LN(W) wartość logarytmu naturalnego wyrażenia W;

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

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

Wywołanie funkcji standardowej LN(W) dla W ≤ 0 oraz funkcji standardowej SQRT(W) dla W < 0 powoduje przerwanie wykonywania programu. Argumenty funkcji SIN(W) i COS(W) oraz wartości funkcji ARCTAN(W) są podawane w radianach.

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(I+Z*3)
(A-3/I+VU POWER 8)

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

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

Proste wyrażenie arytmetyczne:
U-XU + OMEGA*SUM POWER COS(I+Z*3) /7.394E-8 POWER W[I+2,8] POWER (A-3/I+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)
A+ (IF Q THEN N-1 ELSE N)
IF A LESS 0 THEN A/B ELSE IF B = 0 THEN
   B/A ELSE Z
```

Wyrażenie arytmetyczne zbudowane niepoprawnie:

```
A + IF Q THEN N - 1 ELSE N
SIN(A + X)B
IF A LESS 0 THEN U + V
3 SIN(X)
```

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 wartościach liczbowych wyrażeń pierwotnych (szczegóły podano 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 dopóty, 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 argumenty są typu INTEGER, w przypadku przeciwnym 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 wyrażenie:

A / B * 7 / (P - Q) * V / S

oznacza w zapisie konwencjonalnym

$$((((A\cdot(B^{-1}))\cdot 7)\cdot((P-Q)^{-1}))\cdot V)\cdot(S^{-1})$$

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 podstawą, 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 określają następujące reguły:

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 równa się 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 kolejno od jego strony lewej do prawej, jednak z uwzględnieniem dodatkowych reguł.

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

pierwszy:	POWER
drugi:	*/ DIV
trzeci:	+ -

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

3.3.6. Arytmetyka wielkości typu REAL i INTEGER

Zarówno zmienne typu REAL jak i typu INTEGER mają w maszynie cyfrowej ZAM 41 tę samą reprezentację zmiennoprzecinkową. W związku z tym wartości zmiennych muszą zawierać się odpowiednio w przedziałach:

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

$0.8636E-77 \approx 2^{-256} <$ ABS(wartość zmiennej typu REAL) $< 2^{255} \approx 0.5789E77$

Jeżeli wartość funkcji ABS(wartość zmiennej typu REAL) jest mniejsza od 0.8636E-77, to tej zmiennej przypisuje się wartość 0.

Jeżeli wartość funkcji ABS(wartość zmiennej typu REAL) jest większa od 0.5789E77, to maszyna przerywa realizację programu i sygnalizuje błąd.

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

3.4. Wyrażenia 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 dotyczących wyrażeń arytmetycznych, podanych w punkcie 3.3.3.

3.4.4. Typy

Zmiennym i odwołaniom funkcyjnym, użytym jako pierwotne wyrażenia 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 przypadku przeciwnym relacja ma wartość FALSE. W poniższej tabeli podano interpretację operatorów logicznych: NOT (nie), AND (i), OR (lub), IMPL (implikuje) oraz 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 są wykonywane na ogół od lewej do prawej, z uwzględnieniem następujących dodatkowych reguł:

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

pierwsze: wyrażenia arytmetyczne, zgodnie z punktem 3.3.5.

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

trzecia: NOT
czwarta: AND
piąta: OR
szósta: IMPL
siódma: 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 O THEN N ELSE N+1]
IF AB LESS C THEN E3 ELSE Q [IF W NOTGREATER
                              O THEN 9 ELSE N]
```

3.5.3. Znaczenie

Wyrażenie sterujące jest regułą na określanie etykiety instrukcji (patrz rozdział 4. "Instrukcje"). Reguły obliczania wartości wyrażenia sterującego są w pełni 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. "Deklaracja 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 3.4.2.). 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 nazwano 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 też, 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>

W celu objaśnienia 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; 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 GO TO 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;
      L:  FOR I:=1 STEP 1 UNTIL M DO
            FOR K:= I+1 STEP 1 UNTIL M DO
               BEGIN W := A[I,K];
               L:    A[I,K]:= A[K,I];
                     A[K,I]:= W
               END DLA I ORAZ K
      END KONIEC BLOKU Q
```

Przykład bloku zbudowanego niepoprawnie:

```
BEGIN  INTEGER N; REAL W,V;
   L: INP(0,N,V);
   SWITCH P:= E, E1, E2;
   IF N LESS V THEN GO TO P[N]
      ELSE GOTO L;
   E: W:= N + V;
      GO TO L1;
   E1: W:= 2*N + V;
      GO TO L1;
```

```
    E2: W:= N * V;
L1:END BLOKU Z DEKLARACJA
       W NIEWŁASCIWYM KONTEKSCIE
```

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 przez odpowiednią deklarację (rozdział 5. "Deklaracje") zlokalizowany do tego bloku. Oznacza to, że: (a) obiekt, reprezentowany przez ten identyfikator wewnątrz danego bloku, nie istnieje poza tym blokiem oraz (b) 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 w nim nie zadeklarowane, nie są w nim lokalne, tj. reprezentują te same obiekty wewnątrz danego bloku, co i w bloku bezpośrednio 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, a także instrukcję występującą po warunku cyklu (punkt 4.6. "Instrukcje cyklu") należy traktować tak, jak gdyby były one ujęte w nawiasy BEGIN i END i stanowiły bloki.

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.

4.1.4. Ograniczenie

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, poczynając od lewego wyrażenia, aż do prawego.

4.2.3.2. Oblicza się wartość wyrażenia.

4.2.3.3. Wartość wyrażenia przypisuje się wszystkim zmiennym lewych stron z wyrażeniami indeksowymi o wartościach obliczonych 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 odpowiednie przekształcenie jest wykonywane autoratycznie. Przy przekształceniu wartości wyrażenia W z typu REAL na typ INTEGER otrzymuje się wynik:

ENTIER (W + 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 9 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. Przykład

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

4.4.3. Znaczenie

Instrukcja pusta nie powoduje wykonania żadnej czynności. Można się nią posłużyć w celu umieszczenia dodatkowej etykiety w programie.

4.5. Instrukcje warunkowe

4.5.1. Składnia

⟨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 O THEN N:= N+1
IF V GREATER U THEN E: Q:= N+M
   ELSE GO TO R
IF S LESS O 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 aktualnych 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 przypadku przeciwnym instrukcja ta zostanie pominięta i dalsze działanie rozpocznie się od następnej instrukcji.

4.5.3.2. Instrukcje warunkowe

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

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

oraz

```
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 opiszemy następująco: wartości wyrażeń boolowskich występujących 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

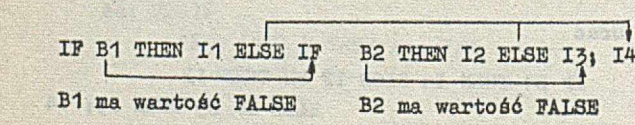
ELSE <instrukcja bezwarunkowa>

jest równoważna konstrukcji

ELSE IF TRUE THEN <instrukcja bezwarunkowa>

W przypadku instrukcji warunkowej drugiego rodzaju, gdy żadne z wyrażeń boolowskich występujących w warunkach nie ma wartości TRUE, wówczas wynik wykonania takiej instrukcji warunkowej sprowadza się jedynie do efektów spowodowanych obliczaniem wyrażeń boolowskich.

Do dalszych wyjaśnień może być pożyteczny schemat:



4.5.4. Skok do wnętrza instrukcji warunkowej

Wynik wykonania instrukcji skoku, prowadzącej do wnętrza instrukcji warunkowej, wypływa 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>
```

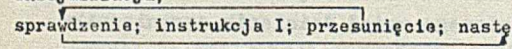
4.6.2. Przykłady

```
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 (patrz punkt 4.1.3) zero lub więcej razy. Ponadto warunek cyklu przypisuje sterowanej przez siebie zmiennej kolejne wartości.

Proces ten można wyjaśnić za pomocą następującego schematu:



W powyższym 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 przypadku przeciwnym wykonuje się instrukcję napisaną po warunku cyklu.

4.6.4. Elementy listy cyklu

Lista cyklu jest regułą, według której otrzymuje się 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. W y r a ż e n i e a r y t m e t y c z n e

Element tego rodzaju daje tylko jedną wartość, mianowicie wartość wyrażenia arytmetycznego, obliczoną bezpośrednio przed odpowiadającym temu elementowi 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 w sposób następujący:

```
      V1:= V:= A;
      V2:= B;
L1:   IF (V1-C)*SIGN(V2) GREATER 0
          THEN GOTO ELEMENTWYCZERPANO;
      instrukcja I;
      V2:= B;
      V1:= V:= V+V2;
      GO TO L1;
```

gdzie V jest zmienną sterowaną przez warunek cyklu, a etykieta ELEMENTWYCZERPANO prowadzi do obliczenia następnego elementu listy cyklu lub - jeżeli rozważany element jest ostatni na liście - do następnej instrukcji programu; V1 i V2 są dodatkowymi zmiennymi roboczymi.

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 w sposób następujący:

```
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 zawsze 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 sama, jaka była bezpośrednio przed wykonaniem instrukcji skoku. Jeżeli natomiast wyjście z instrukcji I było spowodowane wyczerpaniem listy cyklu, to w przypadku, gdy ostatni element listy cyklu był

(a) wyrażeniem arytmetycznym, wówczas zmienna sterowana ma jego wartość;

(b) elementem postaci postępu arytmetycznego bądź elementem "podczas gdy", wówczas zmienna sterowana ma wartość równą pierwszej wartości tej zmiennej powodującej przejście do następnej instrukcji programu.

4.6.6. Skok do wnętrza instrukcji cyklu

Wynik wykonania instrukcji skoku znajdującej się na zewnątrz instrukcji cyklu, a odno-

szącej 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>

Patrz także punkt 3.2.1.

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 są wykonywane 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 przypi-

sań jest taki, jak gdyby treść procedury objęto dodatkowym blokiem i w tym bloku wykonano przypisanie wartości zmiennym lokalnym w nim, zgodne co do typu z odpowiednimi 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, zastępuje się 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 poprzednio, są usuwane przez odpowiednią zmianę identyfikatorów parametrów 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 włączonymi do treści procedury przez proces zamiany a identyfikatorami, których deklaracje obowiązują w miejscu występowania instrukcji procedury lub odwołania funkcyjnego,

są usuwane przez odpowiednią 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 (punkt 5.4.1.) ustala się następująco: lista parametrów aktualnych instrukcji procedury powinna mieć tyle pozycji, ile ma lista parametrów formalnych nagłówka deklaracji procedury; odpowiedniość otrzymuje się przez zestawienie kolejnych par pozycji obu tych list.

4.7.5. Ograniczenia

Aby instrukcja procedury była określona, działania na treści procedury, zdefiniowane w punktach 4.7.3.1. oraz 4.7.3.2, powinny prowadzić do poprawnej instrukcji 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 procedur wyjścia (punkt 6.1. "Instrukcje wyjścia") lub w odwołaniu funkcyjnym, to ten tekst może być użyty w treści procedury jedynie jako parametr aktualny przy wywoływa-

niu dalszych procedur. Ostatecznie, tekstem mogą się posługiwać tylko 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). Ponadto zmienna ta musi być takiego samego typu jak specyfikowany parametr formalny.

4.7.5.3. Parametrowi formalnemu, który w treści procedury jest identyfikatorem tablicy, może odpowiadać tylko taki parametr aktualny, który jest identyfikatorem tablicy o tych samych wymiarach i typie określonym przez specyfikację. Jeżeli parametr formalny jest wymieniony na liście parametrów wywoływanych przez wartość, to lokalna tablica, która pojawi się w treści procedury w wyniku przypisania, otrzyma te same granice indeksów, co i tablica 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. Ograniczenie

Liczba parametrów formalnych procedury nie może przekraczać 127.

4.7.7. 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, otrzymana z objaśniających ograniczników, jest w istocie zbędna.

5. DEKLARACJE

Za pomocą deklaracji określa się pewne właściwości wielkości używanych w programie i wiąże je z identyfikatorami. Deklaracja identyfikatora obowiązuje w jednym bloku. Poza tym blokiem można używać tego identyfikatora 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 włas-

nych") 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 danego bloku więcej niż raz.

Składnia:

<deklaracja> ::= <deklaracja zmiennych prostych>|<deklaracja tablic>|<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
REAL X
```

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, w którym może wystąpić zmienna typu REAL, może również wystąpić zmienna typu INTEGER.

Znaczenie deklaratora OWN omówiono na początku tego rozdziału.

5.2. Deklaracje tablic

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

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

<deklaracja tablic> ::= ARRAY <lista tablic> |
<typ lokalny lub własny> ARRAY
<lista tablic>

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

5.2.3.1. O g r a n i c z e n i a i n d e k s ó w

Ograniczenia indeksów dowolnej tablicy są podane 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. W y m i a r y

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

5.2.3.3. T y p y

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

5.2.3.4. S e g m e n t t a b l i c

Segment tablic może określać nie więcej niż 255 tablic.

5.2.4. Wyrażenia występujące jako ograniczenia indeksów.

5.2.4.1. Wyrażenia te oblicza 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 tablic. Stąd wynika, że w najbardziej zewnętrznym bloku programu w deklaracjach tablic mogą występować tylko stałe ograniczenia indeksów.

5.2.4.3. Tablica jest określona tylko wtedy, gdy wartości wszystkich górnych ograniczeń indeksów nie są mniejsze od wartości odpowiednich dolnych ograniczeń.

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

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

5.2.5. Identyczność zmiennych indeksowanych.

Identyczność zmiennych indeksowanych nie ma związku z ograniczeniami indeksów podanymi w deklaracji tablic. Wartości odpowiednich zmiennych indeksowanych są w dowolnej chwili określone tylko dla tych indeksów, które są zawarte w ograniczeniach obliczonych ostatnim razem.

5.3. Deklaracje 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 liczy się poczynając od lewej strony listy 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 tego elementu na liście, którym jest dane 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, usuwa się zmieniając odpowiednio te ostatnie identyfikatory.

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 MAJACY NAJWIEKSZA WARTOSC BEZ-
    WZGLEDNA POSYLA SIE DO Y. INDEKSY TEGO
    ELEMENTU UMIESZCZA 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 proce-

dury. Zasadniczą częścią składową tej deklaracji jest instrukcja, czyli część 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 spotykane w treści procedury i reprezentujące parametry formalne. W chwili wywołania procedury (punkty 3.2. "Odwołanie funkcyjne" oraz 4.7. "Instrukcje procedury") parametrom formalnym w treści tej procedury 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 też nie. Te z nich, które nie są lokalne w treści procedury, mogą być lokalne w bloku, na którego początku 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, aby deklaracja procedury określała wartość odwołania funkcyjnego, w treści procedury powinna znajdować się co najmniej jedna instrukcja przypisania, w której nazwa procedury występuje z lewej strony symbolu przypisania. Po wywołaniu procedury musi być wykonana co najmniej jedna taka instrukcja. Wartość odwołania funkcyjnego określa ostatnia wykonana spośród wymienionych instrukcji. Z wartości tej będzie się korzystać podczas dalszego obliczania wyrażenia, w którym występuje odwołanie funkcyjne. 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. Specyfikacja

Do nagłówka procedury należy włączyć zbiór specyfikacji, w którym podaje się 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>

6.1.2. Przykłady

Instrukcje wyjścia liczbowego:

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

LICZBA	FORMAT LICZBOWY					
	Y+DDD.DD	Z+DDD.DD	Y-DDD.DD	Z-DDD.DD	YDDD.DD	ZDDD.DD
	1234567	1234567	1234567	1234567	123456	123456
-955	-955.00	-955.00	-955.00	-955.00	-955.00	-955.00
-22	-022.00	-22.00	-022.00	-22.00	-022.00	-22.00
0	+000.00	+0.00	000.00	0.00	000.00	0.00
5	+005.00	+5.00	005.00	5.00	005.00	5.00
631	+631.00	+631.00	631.00	631.00	631.00	631.00
4172183	+417.22E4	+417.22E4	417.22E4	417.22E4	417.22E4	417.22E4
-534362,3235	-534.36E3	-534.36E3	-534.36E3	-534.36E3	-534.36E3	-534.36E3
-19600,44100	-196.00E2	-196.00E2	-196.00E2	-196.00E2	-196.00E2	-196.00E2
-925,62487	-925.62	-925.62	-925.62	-925.62	-925.62	-925.62
-17,292033	-017.29	-17.29	-017.29	-17.29	-017.29	-17.29
-9,722498	-009.71	-9.71	-009.71	-9.71	-009.71	-9.71
-0,8530203	-000.85	-0.85	-000.85	-0.85	-000.85	-0.85
-0,0998504	-000.10	-0.10	-000.10	-0.10	-000.10	-0.10
0,2525252	+000.25	+0.25	000.25	0.25	000.25	0.25
0,8884878	+000.89	+0.89	000.89	0.89	000.89	0.89
7,019692	+007.02	+7.02	007.02	7.02	007.02	7.02
29,2526247	+029.25	+29.25	029.25	29.25	029.25	29.25
413,133939	+413.13	+413.13	413.13	413.13	413.13	413.13
17584,679526	+175.85E2	+175.85E2	175.85E2	175.85E2	175.85E2	175.85E2
8786230,2831	+878.62E4	+878.62E4	878.62E4	878.62E4	878.62E4	878.62E4

LICZBA	FORMAT LICZBOWY					
	Y+BBDDD	Z+DBDDD	Z-.DDDBDD	Y+D.DDDBDD	Z-BD.DDB	ZDBDDDBDDD
	123456	123456	12345678	123456789	1234567	123456789
-955	-0 955	- 955	-.955 00E3	-9.550 00E2	- 9.55 E2	-955
-22	-0 022	-22	-.220 00E2	-2.200 00E1	- 2.20 E1	-22
0	+0 000	+0	.000 00	+0.000 00	0.00	0
5	+0 005	+5	.500 00E1	+5.000 00	5.00	5
631	+0 631	+ 631	.631 00E3	+6.310 00E2	6.31 E2	631
4172183	+4 172E3	+4 172E3	.417 22E7	+4.172 18E6	4.17 E6	4 172 183
-534362,3235	-5 344E2	-5 344E2	-.534 36E6	-5.343 62E5	- 5.34 E5	-53 4362
-19600,44100	-1 960E1	-1 960E1	-.196 00E5	-1.960 04E4	- 1.96 E4	-1 9600
-925,62487	-0 926	- 926	-.925 62E3	-9.256 25E2	- 9.26 E2	-926
-17,292033	-0 017	-17	-.172 92E2	-1.729 20E1	- 1.73 E1	-17
-9,7122498	-0 010	-10	-.971 22E1	-9.712 25	- 9.71	-10
-0,8530203	-0 001	-1	-.853 02	-0.853 02	- 0.85	-1
-0,0998564	-0 000	-0	-.099 86	-0.099 86	- 0.10	-0
0,2525252	+0 000	+0	.252 53	+0.252 53	0.25	0
0,8884878	+0 001	+1	.888 49	+0.888 49	0.89	1
7,019692	+0 007	+7	.701 97E1	+7.019 69	7.02	7
29,2526247	+0 029	+29	.292 53E2	+2.925 26E1	2.93 E1	29
413,133939	+0 413	+ 413	.413 13E3	+4.131 34E2	4.13 E2	413
17584,679526	+1 758E1	+1 758E1	.175 85E5	+1.758 47E4	1.76 E4	17 585
8786230,2831	+8 786E3	+8 786E3	.878 62E7	+8.786 23E6	8.79 E6	8 786 230

LICZBA	FORMAT LICZBOWY				
	ZB+B,DDD	E-DDBDDDED	E+DD.DDDBE-D	E-.DDDBDDDBDDE+DD	ED,DDDBDE+D
	1234567	123456789	12345678901	1234567890123456	1234567890
-955	-9,550E2	-95 500E-2	-95.500 E 1	-.955 000 00E +3	-9.55 00E+2
-22	-2,200E1	-22 000E-3	-22.000 E 0	-.220 000 00E +2	-2.20 00E+1
0	+0,000	00 000E0	+00.000 E 0	.000 000 00E +0	0.000 0E+0
5	+5,000	50 000E-4	+50.000 E-1	.500 000 00E +1	5.000 0E+0
631	+6,310E2	63 100E-2	+63.100 E 1	.631 000 00E +3	6.310 0E+2
4172183	+4,172E6	41 722E2	+41.722 E 5	.417 218 30E +7	4.172 2E+6
-534362,3235	-5,344E5	-53 436E1	-53.436 E 4	-.534 362 32E +6	-5.34 36E+5
-19600,44100	-1,960E4	-19 600E0	-19.600 E 3	-.196 004 41E +5	-1.96 00E+4
-925,62487	-9,256E2	-92 562E-2	-92.562 E 1	-.925 624 87E +3	-9.25 62E+2
-17,292033	-1,729E1	-17 292E-3	-17.292 E 0	-.172 920 33E +2	-1.72 92E+1
-9,7122498	-9,712	-97 122E-4	-97.122 E-1	-.971 224 98E +1	-9.71 22E+0
-0,8530203	-0,853	-85 302E-5	-85.302 E-2	-.853 020 30E +0	-8.53 02E-1
-0,0998564	-0,100	-99 856E-6	-99.856 E-3	-.998 564 00E -1	-9.98 56E-2
0,2525252	+0,253	25 253E-5	+25.253 E-2	.252 525 20E +0	2.525 3E-1
0,8884878	+0,888	88 849E-5	+88.849 E-2	.888 487 80E +0	8.884 9E-1
7,019692	+7,020	70 197E-4	+70.197 E-1	.701 969 20E +1	7.019 7E+0
29,2526247	+2,925E1	29 253E-3	+29.253 E 0	.292 526 25E +2	2.925 3E+1
413,133939	+4,131E2	41 313E-2	+41.313 E 1	.413 133 94E +3	4.131 3E+2
17584,679526	+1,758E4	17 585E0	+17.585 E 3	.175 846 80E +5	1.758 5E+4
8786230,2831	+8,786E6	87 862E2	+87.862 E 5	.878 623 03E +7	8.786 2E+6

LICZBA	STANDARTOWY FORMAT LICZBOWY		
	I	Z	E
	12345678901	12345678901234567	12345678901234567890123 4
-955	- 000955	-955.000000	-9.550 000 000E +2
-22	- 000022	-22.000000	-2.200 000 000E +1
0	000000	0.000000	+0.000 000 000E +0
5	000005	5.000000	+5.000 000 000E +0
631	000631	631.000000	+6.310 000 000E +2
4172183	417218E1	417218.300000E1	+4.172 183 000E +6
-534362,3235	- 534362	-534362,323502	-5.343 623 235E +5
-19600,44100	- 019600	-19600.441000	-1.960 044 100E +4
-925,62487	- 000926	-925.624870	-9.256 248 700E +2
-17,292033	- 000017	-17.292033	-1.729 203 300E +1
-9,7122498	- 000010	-9.712250	-9.712 249 800E +0
-0,8530203	- 000001	-0.853020	-8.530 203 000E -1
-0,0998564	- 000000	-0.099856	-9.985 640 000E -2
0,2525252	000000	0.252525	+2.525 252 000E -1
0,8884878	000001	0.888488	+8.884 878 000E -1
7,019692	000007	7.019692	+7.019 692 000E +0
29,2526247	000029	29.252625	+2.925 262 470E +1
413,133939	000413	413.133939	+4.131 339 390E +2
17584,679526	017585	17584.679526	+1.758 467 953E +4
8786230,2831	878623E1	878623.028308E1	+8.786 230 283E +6

Instrukcje wyjścia tekstowego:

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

Instrukcje wyjścia znakowego:

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

Instrukcje rozmieszczenia:

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

6.1.3. Znaczenie

Procedury wyjścia powodują wyprowadzenie z maszyny cyfrowej danych liczbowych, logicznych, tekstowych lub pojedynczych symboli.

Pierwszy parametr instrukcji procedury wyjścia (instrukcji wyjścia) określa numer wyjścia symbolicznego, używany podczas wykonywania procedury. Definicje numerów wyjścia symbolicznego należy podać w czołówce JOM (patrz Maszyna ZAM 41, Oprogramowanie (System 141), Tom I oraz przykłady w rozdz. 10).

Drugi parametr instrukcji, procedury wyjścia liczbowego, logicznego czy też tekstowego określa 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 tablicy. Wyprowadzenie elementów tablicy jest równoważne wykonaniu poniższego programu.

Przyjmijmy, że do tablicy C odnosiła się deklaracja:

```
ARRAY C [LO:UO,L1:U1,L2:U2,...,LK:UK]
```

wówczas jest realizowany 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 IO:=LO STEP 1 UNTIL UO DO
           OUT (N, < format liczbowy > ,
             C[IO,I1,I2,...,IK])
```

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

6.1.4. Instrukcje wyjścia liczbowego

6.1.4.1. S k ł a d n i a

< instrukcja wyjścia liczbowego > ::= OUT
(< numer wyjścia symbolicznego > < 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 tablicy>
<numer wyjścia symbolicznego> ::= <wyrażenie arytmetyczne>
<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 formatu liczbowego

'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 listy elementów wyjścia liczbowego.

Symbole używane do tworzenia formatu całkowitego, ułamkowego i wykładniczego mają następujące znaczenie:

Symbol	Znaczenie
D	wypisanie cyfry dziesiętnej
B	utworzenie 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 liter B (lub D) o takiej liczbie liter, 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 postaci 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 ustalono standardowe formaty liczbowe - składające się z jednej litery - o następującym znaczeniu:

Format standardowy	Format równoważny
'E'	'E3B+D.3DB3DB3DE+2D3B'
'Y'	'Y3B-6D.6D'
'Z'	'Z3B-6D.6D'
'I'	'Y3B-B6D'

6.1.4.4. O g r a n i c z e n i a

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.

6.1.4.5. U z u p e ł n i e n i a

Jeżeli wartość wyprowadzanej liczby dziesiętnej przekracza przedział wyznaczony przez

zadany format ułamkowy, to będzie ona wyprowadzona według odpowiadającego mu formatu wykładniczego.

Tabelki przedstawione na stronach ALG 6-2 - ALG 6-5 mają zilustrować postać graficzną wyprowadzanych liczb dziesiętnych.

Wypisane w pierwszym wierszu tabelki ciągi cyfr określają liczbę pozycji zajmowanych na arkuszu wydawniczym w przypadku, gdy wartość wyprowadzanej liczby nie przekracza wartości wyznaczonej przez format liczbowy.

6.1.5. Instrukcje wyjścia logicznego

6.1.5.1. S k ł a d n i a

<instrukcja wyjścia logicznego> ::= OUT
(<numer wyjścia symbolicznego> <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 tablicy>

<składowa F> ::= 5F|F

<format logiczny> ::= 'L <składowa B> <składowa F> <składowa B>'

6.1.5.2. P r z y k ł a d y f o r m a t u l o g i c z n e g o

'L3B5FB'
'LBF2B'

'L5F4B'
'LF2B'
'L5F'
'LF'

6.1.5.3. Znaczenie formatu logicznego

Format logiczny określa szablon, według którego są wyprowadzone 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 wówczas, gdy wyprowadzana wielkość ma wartość FALSE, albo T, gdy wyprowadzana wielkość ma wartość TRUE.

6.1.6. Instrukcje wyjścia tekstowego

6.1.6.1. Składnia

<instrukcja wyjścia tekstowego> ::= OUT (<numer wyjścia symbolicznego> <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>

6.1.6.2. Z n a c z e n i e

Procedura wyjścia tekstowego powoduje wyprowadzenie tekstowych elementów wyjścia. Przy wypisywaniu pomija się symbole "'" ograniczające poszczególne teksty.

W tekstach występujących na liście tekstowych elementów wyjścia, symbol ":" jest używany jedynie w celu zmiany sensu symbolu występującego za nim bądź za nim i za replikatorem. (Znaczenie replikatora omówiono w punkcie 6.1.4.3). Symbole te mają wówczas następujące znaczenie:

Symbol	Znaczenie
: B	spacja
: /	przejście do początku następnej linii
: *	przejście do początku następnej strony
: O[1/]	wypisanie symbolu "'"
: Z	wypisanie symbolu "'"
: D	wypisanie symbolu ":"

6.1.7. Instrukcje wyjścia znakowego

6.1.7.1. S k ł a d n i a

<instrukcja wyjścia znakowego> ::= OUTCHAR (<numer wyjścia symbolicznego><ogranicznik parametru><znakowy element wyjścia>)

<znakowy element wyjścia> ::= <wyrażenie arytmetyczne>

1/ litera O

6.1.7.2. Znaczenie

Procedura wyjścia znakowego powoduje wyprowadzenie symbolu, którego kod jest określony wartością wyrażenia arytmetycznego, będącego znakowym elementem wyjścia. (Patrz Dodatek B).

6.1.8. Instrukcje rozmieszczenia

6.1.8.1. Składnia

<instrukcja rozmieszczenia> ::= OUT(<numer wyjścia symbolicznego> <ogranicznik parametru> <format rozmieszczenia>)|OUT(<numer symbolicznego wyjścia> <ogranicznik parametru> <format rozmieszczenia> <ogranicznik parametru> <replikator rozmieszczenia>)

<replikator rozmieszczenia> ::= <wyrażenie arytmetyczne>

<wskaźnik rozmieszczenia> :: <replikator> <symbol rozmieszczenia>|<wskaźnik rozmieszczenia> <replikator> <symbol rozmieszczenia>

<format rozmieszczenia> ::= ' <wskaźnik rozmieszczenia> '

<symbol rozmieszczenia> ::= B | / | *

6.1.8.2. Przykłady formatu rozmieszczenia

'5/15BB'
'//7B'
'2/7B'

6.1.8.3. Znaczenie formatu rozmieszczenia

Format rozmieszczenia określa sposób rozmieszczenia wyników na arkuszu wydawniczym.

Znaczenia symboli B, / i * podano w punkcie 6.1.6.2. Trzeci parametr procedury rozmieszczenia, jeśli występuje, określa liczbę powtórzeń formatu rozmieszczenia.

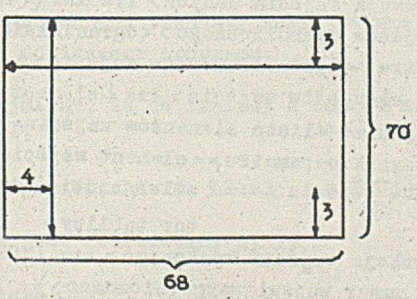
6.1.9. Arkusz wydawniczy

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

liczba symboli w wierszu - 68,
liczba wierszy na stronie - 70,

przy czym marginesy wynoszą:

lewy margines - 4 symbole,
prawy margines - 0 symboli,
górny margines - 3 wiersze,
dolny margines - 3 wiersze.



Drukowanie wyników rozpoczyna się od lewego marginesu. W przypadku przekroczenia przez wyprowadzaną wielkość prawego marginesu, cała jednostka jest automatycznie przenoszona do

następnego wiersza, względnie na nową stronę.

Każdą stronę rozpoczyna ciąg 69 symboli "-" wypisanych w pierwszym wierszu. W następnym wierszu jest umieszczany kolejny numer strony. Pierwsze wykonanie procedury OUT albo OUTCHAR w danym programie powoduje wypisanie numeru strony 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 wejścia symbolicznego⟩⟨ogranicznik parametru⟩⟨lista elementów wejścia⟩)

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

⟨element wejścia⟩ ::= ⟨zmienna⟩|⟨identyfikator tablicy⟩

⟨instrukcja wejścia znakowego⟩ ::= INCHAR(⟨numer wejścia symbolicznego⟩)

⟨numer wejścia symbolicznego⟩ ::= ⟨wyrażenie arytmetyczne⟩

6.2.2. Znaczenie

Procedury wejścia powodują wprowadzenie z wejścia danych liczbowych, logicznych lub pojedynczych symboli kodu. Pierwszy parametr instrukcji procedury wejścia (instrukcji wejścia) określa numer wejścia symbolicznego. Definicje numerów wejścia symbolicznego należy podać w czołówce JOM (patrz Maszyna ZAM 41, Oprogramowanie (System 141), Tom I).

6.2.3. Instrukcja wejścia danych

Procedura wejścia danych powoduje wprowadzenie do maszyny cyfrowej wartości liczbowych lub logicznych i przypisanie tych wartości identyfikatorom, występującym jako dalsze parametry instrukcji. Każdemu elementowi 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 tablicy, to wprowadzenie danych liczbowych określających jej elementy jest równoważne wykonaniu poniższego programu.

Przyjmijmy, że do tablicy C odnosi się deklaracja:

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

Wówczas jest realizowany 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 IO:=LO STEP 1 UNTIL UO DO INP(N,
                 C[IO,I1,I2,...,IK])
```

gdzie N jest numerem wejścia symbolicznego,

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 symbol różny od ":" oraz "=" >

<komentarz> ::= <komentarz otwarty> :| <komentarz otwarty> =

6.2.3.2. Przykład danych liczbowych

```
WYMIARY MACIERZY A
WIERSZY: 4;       KOLUMN: 3;
WYMIARY MACIERZY B
WIERSZY: 3;       KOLUMN: 5;
ELEMENTY MACIERZY A:
 1, 2, 5, 1,
-1, 1, 7, 4,
 1, 0, 1, 3;
```

```
ELEMENTY MACIERZY B:
 5, 2, 1,
-1, 2,-1,
 2, 0, 5,
 1, 1, 4,
-1, 1, 4;
```

6.2.3.3. P r z y g o t o w a n i e d a n y c h

6.2.3.3.1. Liczby występujące jako elementy danych liczbowych muszą być przedstawione w postaci określonej w punkcie 2.5.1. Jednakże procedura wejścia danych ignoruje symbole spacji i nowej linii, zatem cyfry oraz symbole "+", "-" mogą być rozdzielane symbolami spacji lub nowej linii.

6.2.3.3.2. Jeżeli grupą danych liczbowych jest tablica, to poszczególne jej elementy oddziela się separatorem ",", a za ostatnim elementem tablicy umieszcza ";". Separator ";" powoduje przejście do wprowadzania wielkości odpowiadającej następnemu elementowi wejścia. W przypadku, gdy brak ";" po ostatnim elemencie tablicy, wówczas jest wprowadzona liczba elementów określona deklaracją tej tablicy.

6.2.3.3.3. Typ wprowadzanego elementu danych liczbowych musi być zgodny z deklaracją elementu wejścia.

6.2.4. Procedura wejścia znakowego

Procedura wejścia znakowego jest funkcją typu INTEGER powodującą wprowadzenie jednego

symbolu z urządzenia wejściowego i nadanie mu wartości zgodnej z tablicą umieszczoną w Dodatku B.

7. URUCHAMIANIE PROGRAMÓW

7.1. Język operacyjny ALGOLu

Każdy program w ALGOLu 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 oddzielać symbolem ";". W szczególności, czołówka może zawierać jedynie symbol ".". Do języka operacyjnego ALGOLu należą następujące zdania:

```
TEKST
KONTROLA
WYDAWNICTWO: < część parametrowa >
WYPROWADZ PROGRAM WYNIKOWY
PROGRAM WYNIKOWY
```

Ponadto po ograniczniku END kończącym program musi następować symbol "*".

7.1.1. Zdanie TEKST

Zdanie TEKST powoduje wyprowadzenie tekstu programu źródłowego za pomocą wyjścia symbo-

licznego numer 0 translatora, określonego w czołówce JOM (patrz Maszyna ZAM 41, Oprogramowanie (System 141), Tom I). Z przeznaczenia tego zdania wynika, że jest rozsądne wyjściu symbolicznemu numer 0 przypisywać drukarkę wierszową. Wyprowadzany tekst programu źródłowego jest przedzielony co dziesięć wierszy napisem: LINIA <liczba całkowita>, gdzie liczba całkowita jest numerem wiersza programu, następującego po tym napisie. Wiersze programu źródłowego są numerowane począwszy od liczby 0. Liczba wierszy programu nie może przekraczać 4095.

7.1.2. Zdanie KONTROLA

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

7.1.3. Zdanie WYDAWNICTWO

7.1.3.1. S k ł a d n i a

<część parametrowa> ::= <składowa części parametrowej> | <część parametrowa> , <składowa części parametrowej>

<składowa części parametrowej> ::= <parametr 1> = (<parametr 2> , <parametr 3>)

<parametr 1> ::= <numer wyjścia symbolicznego>

<parametr 2> ::= <liczba znaków w wierszu>

<parametr 3> ::= <liczba wierszy na stronie>
<numer wyjścia symbolicznego> ::= 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. P r z y k ł a d

```
WYDAWNICTWO: 0=(120, 72),
             1= (65, 30)
```

7.1.3.3. Z n a c z e n i e

Zdanie WYDAWNICTWO określa niestandardowy format arkusza wydawniczego dla tych numerów wyjścia symbolicznego, z których będą w programie korzystać instrukcje procedur OUT i OUTCHAR. Jeżeli w czołówce brak deklaracji wydawnictwa, to program wynikowy będzie korzystał z formatu standardowego. Liczba znaków w wierszu oraz liczba wierszy na stronie zostały ograniczone do 255.

7.1.4. Zdanie WYPROWADZ PROGRAM WYNIKOWY

Zdanie WYPROWADZ PROGRAM WYNIKOWY oznacza żądanie wyprowadzenia programu wynikowego przez to urządzenie wyjściowe, któremu w czołówce JOM przyporządkowano symboliczne wyjście numer 1 translatora. Jeżeli oznacza ono perforator, to otrzymaną taśmę perforowaną można poprzedzić czołówką JOM i następnie wprowadzać ją do maszyny. W przypadku wielokrotnego korzystania z tego samego programu

źródłowego można w ten sposób pomijać jego tłumaczenie.

Wyprodukowany przez maszynę program wynikowy zawiera następującą czołówkę w języku operacyjnym ALGOLu:

PROGRAM WYNIKOWY.

W czołówce JOM jako nazwę translatora należy umieścić słowo ALGOL. W zależności od treści czołówki napisanej w języku operacyjnym ALGOLu następuje tłumaczenie programu źródłowego albo wprowadzanie do maszyny uprzednio przetłumaczonego programu.

7.2. Sygnalizacja błędów

Podczas tłumaczenia programu źródłowego, a także podczas wykonywania programu wynikowego, jest sprawdzana poprawność przetwarzanego tekstu i interpretowanego programu. Po wykryciu błędu informacja określająca typ tego błędu oraz jego położenie w programie źródłowym jest wyprowadzana za pomocą wyjścia symbolicznego numer 0 translatora.

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. Wszystkie wiersze programu w ALGOLu (oprócz czołówki) są numerowane, przy czym pierwszemu wierszowi przypisuje się numer 0.

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

7.2.1. Błędy w czołówce języka operacyjnego ALGOLu

Błędnie napisane zdanie czołówki języka operacyjnego ALGOLu albo brak czołówki przed pierwszym ogranicznikiem BEGIN powoduje wyprowadzenie tekstu:

BLEDNA CZOLOWKA

Dalszy ciąg programu nie jest czytany przez maszynę.

7.2.2. Błędy wykrywane w pierwszym przebiegu tłumaczenia

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

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

BYLY BLEDY W PROGRAMIE ZRODLOWYM

Błędy i przepełnienia list wykrywane i sygnalizowane podczas pierwszego przebiegu translatora podano w Dodatku C.

7.2.3. Błędy wykrywane w drugim przebiegu tłumaczenia

W drugim przebiegu tłumaczenia materiał przygotowany przez pierwszy przebieg jest przetwarzany na program wynikowy. Jeżeli podczas tych czynności zostanie wykryty błąd, to jest wypisywany odpowiedni sygnał, po czym blok, w którym znaleziono ten błąd, jest przeglądany bez wykonywania jakiejkolwiek kontroli poprawności. Natomiast bloki wewnętrzne względem tego bloku są tłumaczone normalnie - wraz z kontrolą poprawności. Jednakże analizy ich tekstów dokonuje się tak, aby nie wykrywać ciągu błędów wynikających z opuszczenia 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 tłumaczenie przebiegało poprawnie pod względem formalnym, przechodzi się do wykonania dalszych czynności określonych w czołówce języka operacyjnego ALGOLu albo w czołówce JOM. W przypadku wykrycia błędu postępuje się analogicznie jak w pierwszym przebiegu.

Błędy i przepełnienia list wykrywane i sygnalizowane podczas drugiego przebiegu translatora podano w Dodatku D.

7.2.4. Błędy wykrywane podczas realizacji programu wynikowego

Po utworzeniu programu wynikowego można przystąpić do jego wykonania. Ponieważ przy-

jęta metoda tłumaczenia nie pozwala wykryć wszystkich błędów (na przykład takich, że klasa i typ parametru aktualnego nie są zgodne z klasą i typem odpowiedniego parametru formalnego), wielu kontroli poprawności dokonuje się podczas wykonywania programu wynikowego.

W przypadku wykrycia błędu - dalsza praca Interpretera staje się niemożliwa. W związku z tym, za pomocą symbolicznego wyjścia numer 0 programu, jest sygnalizowany błąd, a maszyna przechodzi do wykonania następnej instrukcji określonej w czołówce JOM.

Spis błędów i przepełnień sygnalizowanych podczas wykonywania programu wynikowego podano w Dodatku E.

Koniec pracy Interpretera jest sygnalizowany tekstem:

KONIEC PROGRAMU

wyprowadzonym przez to urządzenie, które przypisano symbolicznemu wyjściu numer 0 programu.

8. JĘZYK ALGOL JAKO CZĘŚĆ SYSTEMU OPROGRAMOWANIA SO 141

W tym rozdziale zostały zebrane informacje niezbędne podczas przygotowywania programu w ALGOLu do realizacji za pomocą maszyny ZAM 41 oraz informacje pozwalające ekonomiczniej stosować dostępne środki.

Programy, które mają być realizowane za pomocą maszyny ZAM 41 poprzedza się specjalnym opisem zwanym czołówką problemu. Czołówka problemu zawiera informacje podawane na etapie wprowadzania programów. Czołówkę formułuje się w Języku Operacyjnym Maszyny. Dla wygody programistów, którzy programują w ALGOLu, zamieszczono zasady tworzenia czołówek problemów.

8.1. Ogólne informacje o Translatorze i Interpreterze ALGOLu

Programy Translatora i Interpretera ALGOLu należą do Biblioteki Systemu SO 141 zapisanej na taśmie magnetycznej. W katalogu bibliotecznym mają przypisaną wspólną nazwę ALGOL.

Translator ALGOLu wprowadza program źródłowy z zewnętrznych nośników informacji za

pomocą wejścia symbolicznego o numerze 0 (patrz punkt 8.3.1.2.) i tłumaczy go na program wynikowy. Program wynikowy jest przedstawiony w specjalnym języku makrorozkazów[1].

Podczas tłumaczenia program wynikowy dzielony jest na segmenty i umieszczany w pamięci bębnowej. Równocześnie są tam umieszczane parametry opisujące strukturę tworzonego programu wynikowego.

Realizacja utworzonego programu wynikowego odbywa się za pomocą Interpretera ALGOLu. Podczas wykonywania programu wynikowego Interpreter tworzy w pamięci operacyjnej programowany stos, na którym są rozmieszczane dane wejściowe, wyniki pośrednie i końcowe oraz informacje opisujące dynamiczną strukturę realizowanego programu. Stos jest również segmentowany i, w przypadku braku miejsca w pamięci operacyjnej, odpowiednie jego segmenty są umieszczane czasowo w pamięci bębnowej.

Warto tutaj zauważyć, że dla każdej zmiennej prostej bądź zmiennej indeksowanej rezerwuje się na stosie dwa słowa maszyny niezależnie od nadanego tym zmiennym typu (REAL, INTEGER, BOOLEAN) za pomocą deklaracji zmiennych prostych bądź deklaracji tablic.

Rozmieszczenie segmentów programu wynikowego i programowanego stosu w pamięci operacyjnej dokonuje się dynamicznie podczas wykonywania programu wynikowego. Suma długości progra-

---

[1] Definicja tego języka znajduje się w książce: B. Randell i L.J. Russell - ALGOL 60 Implementation, AP, London and New York 1964

mu wynikowego i tworzonego stosu nie może przekraczać 32K (tj. 32 x 1024) słów maszyny.

Poniżej określamy środki umożliwiające funkcjonowanie Translatora i Interpretera ALGOLu.

8.1.1. Zestaw maszyny

Translator i Interpreter ALGOLu mogą pracować w zestawie maszyny, w którym zapewnia się do dyspozycji programu użytkowego:

- pamięć operacyjną o pojemności co najmniej 7,5 K słów maszyny;
- pamięć bębnową o pojemności 10 K ÷ 37 K słów maszyny (wymagana pojemność pamięci bębnowej zależy od programu źródłowego);
- współpracę z urządzeniami zewnętrznymi, wśród których znajduje się co najmniej jedno urządzenie czytające (czytnik taśmy papierowej, czytnik kart) i jedno urządzenie piszące (perforator taśmy papierowej, drukarka wierszowa).

8.1.2. Wprowadzanie informacji wejściowych

Translator ALGOLu wprowadza program źródłowy albo gotowy program wynikowy wraz z opisującymi go parametrami (patrz punkty 7.1.3. "Zdanie WYPROWADZ PROGRAM WYNIKOWY" i 10.4. "Przykład 4") za pomocą wejścia symbolicznego o numerze 0.

Interpreter ALGOLu wprowadza za pomocą wejścia symbolicznego o numerze 0[1] dane wejściowe (o ile takie są wymagane) potrzebne podczas realizacji programu wynikowego.

Translator ALGOLu interpretuje symbole, wprowadzane z zewnętrznych nośników informacji, zgodnie z ustalonym dla maszyny ZAM 41 kodem wewnętrznym KW6 (patrz Maszyna ZAM 41, System Oprogramowania SO 141, Opisy Funkcjonalne,(materiały aktualizowane), KODY (4)). W oparciu o ten kod wewnętrzny działa również Interpreter podczas realizacji instrukcji wejść.

Zewnętrzne nośniki informacji:

- 5 kanałowa papierowa taśma perforowana
- karty perforowane
- taśma magnetyczna, na której program źródłowy został uprzednio zapisany za pomocą programu nazwanego Systemem Magazynowania i Aktualizacji Dokumentów (SMAD).

Poniżej podano reprezentacje zewnętrzne dwóch najbardziej rozpowszechnionych kodów stosowanych przy zapisywaniu programów w ALGOLu na zewnętrznych nośnikach informacji. Pierwszy z nich służy do przedstawiania informacji na taśmie papierowej, a drugi - na kartach.

Reprezentacje innych zewnętrznych kodów, dla których są opracowane dekodery (tj. sposo-

---

[1] Interpreter ALGOLu może również współpracować z wejściami symbolicznymi 1 + 7. Z możliwości tej będzie można praktycznie korzystać wówczas, gdy odpowiednie programy obsługujące te wejścia zgodnie z definicjami umieszczonymi w czołówce problemu będą dołączane automatycznie.

by przejścia z symboli kodu zewnętrznego na symbole kodu wewnętrznego, bądź odwrotnie), można znaleźć w pracy: Maszyna ZAM 41, System Oprogramowania SO 141, Opisy Funkcjonalne, (materiały aktualizowane), KODY. Programy w ALGOLu można zapisywać na zewnętrznych nośnikach informacji za pomocą kodów, które reprezentują zbiór symboli wystarczający do przedstawienia symboli podstawowych opisanej reprezentacji konkretnej języka ALGOL.

8.1.2.1. Reprezentacja zewnętrzna dla kodu M2

Nośnikiem informacji dla kodu M2 (Międzynarodowy Kod Dalekopisowy Numer 2) jest 5-kanałowa taśma papierowa. Tablica 8.1. zawiera kod M2 zmodyfikowany, dostosowany do potrzeb zapisywania programów dla maszyn matematycznych.

8.1.2.2. Reprezentacja zewnętrzna dla kodu IBM

Nośnikiem informacji dla kodu IBM są 80-kolumnowe karty perforowane. Wiersze takich kart są oznaczone numerami 12, 11, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Perforacje w poszczególnych kolumnach przedstawiają symbole kodu IBM (patrz Maszyna ZAM 41, System Oprogramowania 141, Opisy Funkcjonalne,(materiały aktualizowane), KODY (7)) zgodnie z tablicą 8.2. W tablicy uwzględniono tylko te symbole, które są symbolami podstawowymi opisanej reprezentacji konkretnej języka ALGOL.

Tablica 8.1.

PERFORACJA	SYMBOL [1] strona liter	strona cyfr
.	NUL* (NU)	
. 0	T (XT)	5 (Z5)
. 0	CR* (CR)	
. 0 0	O (XO)	9 (Z9)
. 0	SP* (SP)	
. 0 0	H (XH)	◊ (N6)
. 0 0	N (XN)	, (CM)
. 0 0 0	M (XM)	. (FP)
0 .	LF* (LF)	
0 . 0	L (XL)	) (RP)
0 . 0	R (XR)	4 (Z4)
0 . 0 0	G (XG)	] (RB)
0 . 0	I (XI)	8 (Z8)
0 . 0 0	P (XP)	0 (Z0)
0 . 0 0	C (XC)	: (CL)
0 . 0 0 0	V (XV)	= (EQ)
0 .	E (XE)	3 (Z3)
0 . 0	Z (XZ)	+ (PL)
0 . 0	D (XD)	CS* (CS)
0 . 0 0	B (XB)	* (AS)
0 . 0	S (XS)	' (AP)
0 . 0 0	Y (XY)	6 (Z6)
0 . 0 0	F (XF)	[(LB)
0 . 0 0 0	X (XX)	/ (SL)
0 0 .	A (XA)	- (MI)
0 0 . 0	W (XW)	2 (Z2)
0 0 . 0	J (XJ)	; (SE)
0 0 . 0 0	'C Y F R Y'	
0 0 . 0	U (XU)	7 (Z7)
0 0 . 0 0	Q (XQ)	1 (Z1)
0 0 . 0 0	K (XK)	((LP)
0 0 . 0 0 0	'L I T E R Y'	

1) Symbole opatrzone * nie reprezentują siebie same - patrz norma ISO R646.

Tablica 8.2

PERFORACJA numery kolumn	SYMBOL[1]		PERFORACJA numery kolumn	SYMBOL[1]	
	BLANK*		11/5	N	(XN)
12	+	(PL)	11/6	O	(XO)
11	-	(MI)	11/7	P	(XP)
0	0	(Z0)	11/8	Q	(XQ)
1	1	(Z1)	11/9	R	(XR)
2	2	(Z2)	0/1	/	(SL)
3	3	(Z3)	0/2	S	(XS)
4	4	(Z4)	0/3	T	(XT)
5	5	(Z5)	0/4	U	(XU)
6	6	(Z6)	0/5	V	(XV)
7	7	(Z7)	0/6	W	(XW)
8	8	(Z8)	0/7	X	(XX)
9	9	(Z9)	0/8	Y	(XY)
12/1	A	(XA)	0/9	Z	(XZ)
12/2	B	(XB)	3/8	=	(EQ)
12/3	C	(XC)	4/8	'	(AP)
12/4	D	(XD)	5/8	:	(CL)
12/5	E	(XE)	12/3/8	.	(FP)
12/6	F	(XF)	12/4/8	)	(RP)
12/7	G	(XG)	12/5/8	[	(LB)
12/8	H	(XH)	11/3/8	SP*	(SP)
12/9	I	(XI)	11/4/8	*	(AS)
11/1	J	(XJ)	11/5/8	]	(RB)
11/2	K	(XK)	11/6/8	;	(SE)
11/3	L	(XL)	0/3/8	,	(CM)
11/4	M	(XM)	0/4/8	(	(LP)

[1] symbole opatrzone * nie reprezentują siebie same - patrz norma ISO R646.

Znak sterujący LF (nowa linia) jest reprezentowany przez koniec karty.

8.1.3. Wyprowadzanie informacji wyjściowych

Translator ALGOLu wyprowadza za pomocą wyjścia symbolicznego o numerze 0 informacje o błędach składniowych wykrytych podczas tłumaczenia lub tekst wprowadzanego programu źródłowego. Za pomocą wyjścia symbolicznego o numerze 1 można wyprowadzić program wynikowy utworzony przez Translator ALGOLu.

Rezultaty obliczeń[1)] otrzymane podczas realizacji programu wynikowego, a także informacje o wykrytych błędach wyprowadza Interpreter za pomocą wyjścia symbolicznego o numerze 0.

Symbole do wyprowadzenia są podawane podprogramom obsługującym wyjścia symboliczne w kodzie wewnętrznym KW6.

8.1.4. Zasady współdziałania Systemu Operacyjnego z Translatorem i Interpreterem ALGOLu

Translator ALGOLu funkcjonuje w Systemie Operacyjnym 141 na prawach translatora, Interpreter - na prawach programu użytkowego.

Wśród informacji pozostawianych Systemowi Operacyjnemu przez Translator ALGOLu po utwo-

---

1) Wyniki obliczeń można będzie wyprowadzać za pomocą wyjść symbolicznych o numerach 1 ÷ 7 wówczas, gdy odpowiednie podprogramy obsługujące te wyjścia zgodnie z definicjami umieszczonymi w czołówce problemu będą dołączane automatycznie.

rzeniu programu wynikowego znajduje się żądanie automatycznego zainicjowania pracy Interpretera ALGOLu w przypadku wywołania programu wynikowego.

8.2. Uwagi o formułowaniu programów w ALGOLu

Przedstawienie programu wynikowego w postaci makrorozkazów realizowanych za pomocą Interpretera ALGOLu powoduje, że czas wykonywania takiego programu jest stosunkowo długi. Dlatego programy należy tak formułować, aby unikać wykonywania zbędnych operacji. Poniżej zwrócimy uwagę na dwie typowe sytuacje.

Jedną z czasochłonnych czynności jest określenie położenia zmiennej indeksowanej. W wielu programach można wyeliminować wielokrotne występowanie tej samej zmiennej indeksowanej wprowadzając dodatkowe zmienne proste.

Podczas realizacji treści procedury najkrócej oblicza się wartości parametrów aktualnych, odpowiadające parametrom formalnym umieszczonym na liście parametrów wywoływanych przez wartość. Ta uwaga nie dotyczy tablic wywoływanych przez wartość.

8.3. Praca w systemie SO 141

Translator i Interpreter ALGOLu są elementami systemu oprogramowania SO 141. Efektywność ich pracy - zwłaszcza czas realizacji programu wynikowego utworzonego przez Translator ALGOLu - zależy w dużej mierze od zestawu maszyny użytkowej. Maszynę użytkową ustala Supervisor aktualnie obsługujący maszynę ZAM 41.

Supervisor jest programem, który należy traktować jako nieodłączną część składową maszyny; współdziała on z maszyną i systemem oprogramowania. Supervisor organizuje jednoprogramową lub dwuprogramową pracę maszyny, obsługuje przerwania, realizuje operacje wejścia i wyjścia.

System oprogramowania SO 141 wyposażony jest w kilka Supervisorów. Każdy z nich udostępnia programom inny zestaw maszyny. Wyboru Supervisora dokonuje operator podczas sprowadzania systemu SO 141 z taśmy magnetycznej (patrz Maszyna ZAM 41, System Oprogramowania SO 141, Opisy Funkcjonalne,(materiały aktualizowane), ISC) i inicjowania jego pracy. Wybór Supervisora powinien być uzależniony od potrzeb sformułowanego problemu oraz od rzeczywistego (fizycznego) zestawu maszyny.

Translator i Interpreter ALGOLu uzyskują najlepsze parametry eksploatacyjne (takie jak: czas realizacji programu, różnorodność zewnętrznych nośników informacji) wówczas, gdy maszyna współdziała z Supervisorami organizującymi jednoprogramową pracę dla zestawów o pojemności pamięci operacyjnej 20 K słów maszyny (np.: Supervisor o nazwie SVC).

Z Supervisorów organizujących jednoprogramową pracę maszyny dla zestawów o pojemności pamięci operacyjnej 12 K słów maszyny należy tutaj wyróżnić Supervisor SVA, który zapewnia współpracę z minimalną liczbą urządzeń zewnętrznych (jeden czytnik taśmy papierowej, jeden perforator taśmy papierowej i drukarka wierszowa), ale udostępnia programowi użytkowemu większy obszar pamięci operacyjnej.

8.3.1. Opisywanie problemów

Problemem nazywa się sekwencję czynności realizowaną przez System Operacyjny[1] SO 141 w procesie wprowadzania i wykonywania programów użytkowych. Na przykład podczas wprowadzania i wykonywania programu w ALGOLu należy najpierw, w celu przetłumaczenia programu źródłowego na program wynikowy, wywołać translator ALGOLu, a potem zainicjować jego realizację.

Problemy opisuje się w Języku Operacyjnym Maszyny - JOM. Język Operacyjny Maszyny zawiera zdania, za pomocą których

- opisuje się problem tj. ustala przebieg pracy maszyny oraz określa wejścia i wyjścia symboliczne używane w programach;
- żąda się wyprowadzenia tytułu.

Opis problemu w JOM nazywa się czołówką problemu.

Poniżej przedstawiono krótko zasady tworzenia czołówek problemu. Z opisu Języka Operacyjnego Maszyny (patrz Maszyna ZAM 41, Oprogramowanie (System 141), Tom I) zostały wybrane i omówione te informacje, które pozwolą formułować problemy zawierające zagadnienia zaprogramowane w ALGOLu.

Czołówka problemu składa się z jednego lub kilku zdań JOM. Poszczególne zdania oddzielane są symbolem ";", a ostatnie zdanie kończy się symbolem ".". Kolejność wyszczególniania zdań jest nieistotna.

---

1) Systemem Operacyjnym nazywa się część Systemu Oprogramowania, za pośrednictwem której realizowane jest współdziałanie różnych programów Systemu Oprogramowania.

Przykłady typowych czołówek problemów znajdują się w rozdziale 10.

8.3.1.1. Zdanie zdefiniuj problem (ustalenie przebiegu pracy maszyny)

Ustalenie przebiegu pracy maszyny polega na określeniu kolejności inicjowania programów należących do Biblioteki Systemu lub programów wynikowych (tj. programów otrzymanych w wyniku pracy translatorów) umieszczonych w pamięci bębnowej.

Przebieg pracy maszyny ustala się za pomocą zdania zdefiniuj problem. Składnia zdania zdefiniuj problem jest następująca:

```
<zdefiniuj problem> ::= PROBLEM: <lista instrukcji>
<lista instrukcji> ::= <instrukcja>|<lista instrukcji> ,<instrukcja>
<instrukcja> ::= <wywołanie programu z Biblioteki Systemu>|<wywołanie programu wynikowego>
<wywołanie programu z Biblioteki Systemu> ::= ALGOL|WYBIN|WEBIN|EDAL
<wywołanie programu wynikowego> ::= PROGRAM
```

Lista instrukcji może zawierać najwyżej 8 instrukcji.

Podczas realizacji problemu system operacyjny inicjuje wykonywanie programów zgodnie z kolejnością ich wyszczególnienia na liście instrukcji. W dalszym opisie zdanie zdefiniuj problem jest nazywane instrukcją działania maszyny.

8.3.1.2. Zdania zdefiniuj wejścia-wyjście symboliczne (określenie wejść i wyjść symbolicznych)

Występujące w programach operacje dotyczące wprowadzenia i wyprowadzenia informacji są opatrzone parametrem zwanym odpowiednio wejściem i wyjściem symbolicznym. Wejście czy też wyjście symboliczne jest umownym oznaczeniem reprezentującym zarówno urządzenie zewnętrzne, za pośrednictwem którego ma się odbywać dana operacja wejściowa lub wyjściowa, jak i dekoder tj. sposób przejścia z symboli kodu zewnętrznego na symbole kodu wewnętrznego, bądź odwrotnie. Wejście (wyjście) symboliczne oznacza się liczbą całkowitą bez znaku z przedziału [0,15]. W jednym programie można stosować kilka różnych wejść lub wyjść symbolicznych.

Taki sposób przedstawienia operacji wejścia i wyjścia obowiązuje we wszystkich programach funkcjonujących w systemie, niezależnie od tego czy program współdziała z systemem na prawach translatorów, czy też na prawach programów użytkowych.

Podczas kompletowania problemu należy nadać znaczenia wejściom i wyjściom symbolicznym używanym w programach wyszczególnionych w instrukcji działania maszyny.

Składnia definicji wejścia-wyjścia symbolicznego jest następująca:

<definicja wejścia-wyjścia symbolicznego> ::=
<numer wejścia-wyjścia symbolicznego> =
<nazwa podprogramu>(<parametry podprogramu>)

< parametry podprogramu > ::= < nazwa urządzenia >, < nazwa dekodera >

Definiując wejście (wyjście) symboliczne należy zwracać uwagę, aby nazwa podprogramu (obsługującego wybrane urządzenie zewnętrzne), nazwa urządzenia oraz nazwa dekodera

- wyznaczały odpowiednio podprogram, urządzenie i dekoder znajdujące się w zestawie,
- dotyczyły wprowadzania (dla wejść) albo wyprowadzania (dla wyjść).

Wykazy wszystkich podprogramów obsługujących urządzenia zewnętrzne oraz dekoderów przytoczone są w opracowaniu: Maszyna ZAM 41, System Oprogramowania SO 141, Opisy Funkcjonalne, (materiały aktualizowane), DOD. Nazwy urządzeń wyszczególniono w opisie JOM.

W tablicach 8.3 i 8.4 podano najbardziej typowe definicje wejść i wyjść symbolicznych. Dla skoncentrowania uwagi wszystkie definicje odnoszą się do wejścia (wyjścia) symbolicznego o numerze 0.

Wejścia i wyjścia symboliczne używane w programach są określane za pomocą zdań "zdefiniuj wejścia-wyjścia symboliczne". Składnia zdań zdefiniuj wejścia-wyjścia symboliczne jest następująca:

< zdefiniuj wejścia-wyjścia symboliczne > ::= < określenie grupy wejść-wyjść > : < lista definicji >

< określenie grupy wejść-wyjść > ::= < określenie typu > < określenie rodzaju >

<określenie rodzaju> ::= TRANSLATOROW|
PROGRAMOW| <puste>
<określenie typu> ::= WEJSCIA|WYJSCIA
<lista definicji> ::= <definicja wejścia-
wyjścia symbolicznego>|<lista definicji>,
<definicja wejścia-wyjścia symbolicznego>

Za pomocą zdań zdefiniuj wejścia-wyjścia symboliczne ustala się osobno wejścia (wyjścia) symboliczne dla programów współdziałających z systemem operacyjnym na prawach translatorów i programów współdziałających na prawach programów użytkowych. Jeżeli określenie grupy wejść-wyjść ma postać WEJSCIA albo WYJSCIA, to definicje wejść albo wyjść symbolicznych umieszczone na liście definicji tego zdania dotyczą wszystkich programów współdziałających z systemem operacyjnym podczas realizacji problemu.

Podczas inicjowania problemu zostają ustalone standardowe definicje wejścia i wyjścia symbolicznego o numerze 0. Te definicje dotyczą zawsze Podstawowego Wejścia Symbolicznego Zestawu i Podstawowego Wyjścia Symbolicznego Zestawu. Na przykład dla zestawów określonych przez Supervisory SVA i SVC Podstawowym Wejściem Symbolicznym Zestawu jest czytnik taśmy papierowej CPO, a Podstawowym Wyjściem Symbolicznym Zestawu jest perforator taśmy papierowej DPO. W tych przypadkach wejście i wyjście symboliczne o numerze 0 są definiowane za pomocą zdań:

WEJSCIA: 0 = PCTP (CTP1, DM2C)
WYJSCIA: 0 = PPTP (PTP1, DM2P)

Ta definicja wejścia symbolicznego o numerze 0 obowiązuje do chwili przekazania sterowania pierwszemu programowi wyszczególnionemu w instrukcji działania maszyny użytkowej. Nową definicję wejścia symbolicznego o numerze 0 ustala się wówczas, gdy wystąpiła odpowiednia definicja w czołówce problemu. Natomiast nową definicję wyjścia symbolicznego 0 ustala się w momencie zidentyfikowania odpowiedniej definicji w czołówce problemu.

Tablica 8.3

Definicja Wejścia Symbolicznego	O b j a ś n i e n i a
0 = PCTP (CTP1, DM2C)	Podprogram Czytania Taśmy Papierowej (czytnik taśmy papierowej CP0, Dekoder M2 Czytanie)
0 = PCTP (CTP2, DM2C)	Podprogram Czytania Taśmy Papierowej (czytnik taśmy papierowej CP1, Dekoder M2 Czytanie)
0 = PCKP (CK, DIBM)	Podprogram Czytania Kart Prosty (czytnik kart CK, Dekoder IBM)
0 = PCM (1, D86C)	Podprogram Czytania Monitora (monitor, Dekoder z kodu KW8 na kod KW6)

Tablica 8.4

Definicja Wyjścia Symbolicznego	O b j a ś n i e n i a
0 = PPTP (PTP1, DM2P)	Podprogram Perforowania Taśmy Papierowej (perforator taśmy papierowej DP0, Dekoder M2 perforowanie)
0 = PPTP (PTP2, DM2P)	Podprogram Perforowania Taśmy Papierowej (perforator taśmy papierowej DP1, Dekoder M2 perforowanie)
0 = PDW (DX, DDX1)	Podprogram Drukarki Wierszowej (drukarka DX, Dekoder DX1)
0 = PPM (0, D68P)	Podprogram Pisania na Monitor (monitor - pisanie 0, Dekoder z kodu KW6 na kod KW8)

8.3.1.3. Z d a n i e w y p i s z t y t u ł

Składnia zdania tytuł

<zdanie TYTUL> ::= TYTUL: <tekst nie zawierający ani symbolu ",", ani symbolu "." >

Tekst zostaje wyprowadzony za pośrednictwem wyjścia symbolicznego o numerze 0 bezpośrednio po zidentyfikowaniu zdania TYTUL.

8.4. Edytor ALGOLu

Edytor ALGOLu[1] przedstawia program źródłowy w ALGOLu w postaci akceptowanej przez System Magazynowania i Aktualizacji Dokumentacji - SMAD (patrz Maszyna ZAM 41, System Oprogramowania SO 141, Opisy Funkcjonalne, (materiały aktualizowane)).

Edytor ALGOLu nadaje programowi źródłowemu nową formę graficzną uwypuklającą jego strukturę logiczną i wprowadza podział na strony zgodnie z wymaganiami SMAD, a ponadto sporządza skorowidz wszystkich zadeklarowanych identyfikatorów.

Edytor ALGOLu funkcjonuje w systemie na prawach programu użytkowego. W katalogu bibliotecznym przypisano mu nazwę EDAL.

Informacją wejściową dla programu EDAL jest program źródłowy w ALGOLu (łącznie z poprzedzającą go czołówką w języku operacyjnym ALGOLu i symbolem "*" wskazującym koniec programu) uzupełniony odpowiednio przedstawionymi znakami korektorskimi, które określają sposób numerowania stronic. Informacja wejściowa jest wprowadzana za pomocą wejścia symbolicznego o numerze 0.

Zredagowany tekst programu źródłowego zostaje wyprowadzony za pomocą wyjść symbolicznych o numerach 0 i 1.

Wyjście symboliczne o numerze 0 jest prze-

---

[1] Program Edytor ALGOLu został opracowany przez Irenę Matusiak

znaczone do przedstawienia zredagowanego programu w formie dokumentacyjnej. Tekst jest łamany na dwie szpalty. Bezpośrednio za tekstem programu zostaje wyprowadzony skorowidz zadeklarowanych identyfikatorów. W czołówce problemu, wyjściu symbolicznemu o numerze 0 należy przypisać drukarkę wierszową.

Wyjście symboliczne o numerze 1 jest przeznaczone do wyprowadzenia zredagowanego programu w postaci akceptowanej przez SMAD i Translator ALGOLu.

8.4.1. Przedstawienie symboli korektorskich

8.4.1.1. Składnia symboli korektorskich

<znacznik podziału> ::= W|W = <litera>|W = <cyfra>

<symbol korektorski> ::= <nowa linia> : [<znacznik podziału>] ; <nowa linia>

8.4.1.2. Znaczenie

Symbol korektorski definiuje sposób numerowania stron. Strony są numerowane kolejnymi liczbami naturalnymi z prefiksem ustalonym przez znacznik podziału. Prefiksy są określane według reguły opisanej w tabelce:

znacznik podziału	prefiks
W	S.
W = <litera>	<litera> .
W = <cyfra>	<cyfra> .

8.4.2. Przygotowanie informacji wejściowej dla Translatora EDAL

Pierwszy symbol korektorski należy umieścić przed czołówką w języku operacyjnym ALGOLu.

Symbole korektorskie można również umieszczać w programie źródłowym. W tym przypadku symbol korektorski trzeba poprzedzić słowem zastrzeżonym COMMENT.

8.4.3. Parametry określające stronę

Parametry określające wypełnienie strony określa się podczas pracy programu EDAL za pośrednictwem monitora. Przed rozpoczęciem wprowadzania informacji wejściowej zostaje wypisany następujący tekst na monitorze:

```
PODAJ
O;
```

W odpowiedzi należy podać dwie liczby, z których pierwsza - tzw. parametr 1 - określa liczbę wierszy tekstu na stronie, a druga - tzw. parametr 2 - określa liczbę wszystkich wierszy (długość strony) na stronie. Parametry należy tak dobierać, aby spełniały warunek:

$$11 + \text{parametr1} < \text{parametr2}.$$

Odpowiedź wypisuje się zgodnie z następującym wzorcem:

```
parametr1, parametr2 *
```

Dla kontroli wyprowadza się te parametry wraz z informacjami opisującymi przyczynę zakończenia za pomocą wyjścia symbolicznego O.

8.4.4. Przykład czołówki problemu

```
WEJSCIA PROGRAMOW: O=PCTP(CTP1, DM2C);
WYJSCIA PROGRAMOW: O=PDW(DX,DDX1),
                   1=PPTP(PTP2,DM2P);
PROBLEM: EDAL.
```

9. ZESTAWIENIE RÓŻNIC REPREZENTACJI KONKRETNEJ ALGOLU W STOSUNKU DO JĘZYKA WZORCOWEGO

Język wzorcowy	Zmiana
2.1.	<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
2.2.2.	<wartość logiczna> ::= TRUE| FALSE
2.3.	<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|ARRAY|SWITCH| PROCEDURE
	<specyfikator> ::= STRING|LABEL VALUE

2.3.1. Słowa zastrzeżone. Ograniczniki będące słowami utworzonymi z liter, a także wartości logiczne, są słowami zastrzeżonymi, tzn. nie można ich używać jako identyfikatorów (zastrzeżenie to nie dotyczy separatora E i poszczególnych składowych operatora GO TO; można używać identyfikatorów: GO,TO i E). Wewnątrz słowa zastrzeżonego nie może występować spacja ani nowa linia. W operatorze następstwa GO TO między słowami GO i TO może pojawić się dokładnie jedna spacja, ale nie może występować nowa linia. Jeżeli bezpośrednio po słowie zastrzeżonym nie następuje żaden ogranicznik spośród ograniczników różnych od słów zastrzeżonych, to należy po nim umieścić spację lub nowę linię. Spis słów zastrzeżonych wraz z ich znaczeniem w języku polskim umieszczono w Dodatku A.

Przyjmuje się, że następujące ograniczniki są równoważne:

Ogranicznik	Jego równoważnik
GOTO	GO TO
EQUAL	=

W ograniczniku := między symbolami : i = nie może występować ani spacja, ani nowa linia.

2.3.2. Komentarze. Następujące reguły pozwalają wstawić komentarz między symbolami programu:

Ciąg symboli podstawowych	Jego równoważnik
; COMMENT <ciąg dowolnych symboli nie zawierający ;>;	;
BEGIN COMMENT <ciąg dowolnych symboli nie zawierający ;>;	BEGIN
END <ciąg dowolnych symboli nie zawierający ; ani END, ani ELSE>	END

Równoważność oznacza tu, że translator zastąpi każdą z trzech struktur podanych w lewej kolumnie (nie dotyczy to struktur zawartych w tekstach - patrz p. 2.6. "Teksty") odpowiednim symbolem z prawej kolumny; zastąpienie to nie będzie miało jakiegokolwiek wpływu na program. Przyjmuje się też, że pierwszy (przy czytaniu symboli programu od lewej strony wiersza do prawej) komentarz ma pierwszeństwo w zastępowaniu go równoważnikiem przed strukturami znajdującymi się w dalszym ciągu programu.

Szczególna postać komentarza, mianowicie komentarz rozpoczynający się od słowa COMMENT, po którym bezpośrednio następuje symbol *, może służyć do warunkowego włączenia instrukcji do programu (patrz punkty 7.1. "Język operacyjny ALGOLu" i 7.1.2. "Zdanie KONTROLA").

2.4.3. Pierwszy akapit zastąpiono następującym:

Identyfikatory nie mają samoistnego sensu, lecz służą jedynie do nazywania zmiennych prostych, tablic, etykiet, przełączników i procedur. Można je wybierać dowolnie, jednak z uwzględnieniem ograniczeń wymienionych 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.

2.4.4. Ograniczenie. W jednym programie można deklarować najwyżej 511 różnych identyfikatorów. Maksymalna długość identyfikatora nie może przekraczać 68 liter i (lub) cyfr.

2.5.1. < cecha > ::= E < liczba całkowita >
< liczba bez znaku > ::= < liczba dziesiętna > | < liczba dziesiętna > < cecha >

2.5.3. Dodano następujące zdania:

Wewnątrz liczby nie może występować ani spacja, ani nowa linia. Jeżeli bezpośrednio po liczbie nie następuje żaden ogranicznik spośród ograniczników różnych od słów zastrzeżonych, to należy po niej umieścić spację lub nową linię. Ograniczenia nałożone na liczby podano w punkcie 3.3.6. "Arytmetyka wielkości typu REAL i INTEGER".

2.6.1. Składnia:

<tekst> ::= '<dowolny ciąg symboli nie zawierający '>'

2.6.3. Znaczenie. Teksty stosuje się jako parametry standardowych procedur wyjścia (patrz punkt 6.1. "Instrukcje wyjścia"). Tekst może zawierać najwyżej 384 symbole (wraz z obydwoma symbolami ').

3.1.4.3. Wartość bezwzględna indeksu nie może przekraczać liczby 32767.

3.2.3.1. Ograniczenie. W ograniczniku parametru postaci

)<tekst literowy> :(

nie może występować symbol nowej linii.

3.2.4. Funkcje standardowe. Jeżeli w programie nie występują deklaracje (patrz rozdział 5. "Deklaracje") odnoszące się do wymienionych poniżej identyfikatorów, to wówczas te identyfikatory są nazwami funkcji standardowych ALGOLu:

ABS
ARCTAN
COS
ENTIER
EXP
INCHAR
LN
SIGN
SIN
SQRT

W przypadku, gdy programista nie chce korzystać w programie lub jego części z jakiejś funkcji standardowej, może wtedy używać jej nazwy jako identyfikatora zadeklarowanej w danym programie wielkości.

3.2.4.1. Znaczenie. Poniższe odwołania do funkcji standardowych oznaczają odpowiednio:

ABS(W)	wartość bezwzględną wartości wyrażenia W;
ARCTAN(W)	wartość główną funkcji arcus tangens W;
COS(W)	wartość funkcji cosinus W;
ENTIER(W)	wartość największej liczby całkowitej nie większej od wartości W;
EXP(W)	wartość funkcji wykładniczej wartości W;
INCHAR(W)	wartość symbolu wprowadzanego z urządzenia wejściowego związanego z numerem wejścia symbolicznego W;
LN(W)	wartość logarytmu naturalnego wyrażenia W;
SIGN(W)	znak algebraiczny wartości W (+1 dla W > 0, 0 dla W = 0, -1 dla W < 0);
SIN(W)	wartość funkcji sinus W;
SQRT(W)	wartość dodatniego pierwiastka kwadratowego z W;

Funkcje te są określone zarówno dla argumentów typu REAL, jak i typu INTEGER. Funkcje: ENTIER (W), SIGN(W)

i INCHAR(W) przyjmują wartości typu INTEGER, a pozostałe funkcje standardowe - wartości typu REAL.

Funkcje: ARCTAN(W), COS(W), EXP(W), LN(W), SIN(W) i SQRT(W) są obliczane za pomocą algorytmów dających wyniki z błędem bezwzględnym mniejszym od 2^{-38}.

Wywołanie funkcji standardowej LN(W) dla $W \leq 0$ oraz funkcji standardowej SQRT(W) dla $W < 0$ powoduje przerwanie wykonywania programu. Argumenty funkcji SIN(W) i COS(W) oraz wartości funkcji ARCTAN(W) są podawane w radianach.

3.2.5. Usunięto punkt 3.2.5.

3.3.6. Arytmetyka wielkości typu REAL i INTEGER. Zarówno zmienne typu REAL, jak i typu INTEGER mają w maszynie cyfrowej ZAM 41 tę samą reprezentację zmiennoprzecinkową. W związku z tym wartości zmiennych muszą zawierać się odpowiednio w przedziałach:

$$-2^{38} = -274877906944 < WZI < 274877906944 = 2^{38}$$

$$0.8636E\text{-}77 \approx 2^{-256} < ABS(WZR) < 2^{255} \approx 0.5789E77$$

gdzie WZI oznacza wartość zmiennej typu INTEGER, a WZR - wartość zmiennej typu REAL.

Jeżeli wartość funkcji ABS(WZR) jest mniejsza od 0.8636E-77, to

zmiennej przypisuje się wartość 0. Jeżeli wartość funkcji ABS(WZR) jest większa od 0.5789E77, to maszyna przerywa realizację programu i sygnalizuje błąd.

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

3.5.1. < etykieta > ::= < identyfikator >

3.5.5. Usunięto punkt 3.5.5.

4.1.3. Ostatnie zdanie w drugim akapicie zastąpiono przez: W tym kontekście treść procedury a także instrukcję występującą po warunku cyklu (patrz p. 4.6. "Instrukcje cyklu") należy traktować tak, jak gdyby były one ujęte w nawiasy BEGIN i END i stanowiły bloki.

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

4.3.1. < instrukcja skoku > ::= GO TO < wyrażenie sterujące > | GOTO < wyrażenie sterujące >

4.5.3.2. Czwarty akapit zastąpiono następującym:

W przypadku instrukcji warunkowej drugiego rodzaju, gdy żadne z wyra-

żeń boolowskich występujących w warunkach nie ma wartości TRUE, wówczas wynik wykonania takiej instrukcji warunkowej sprowadza się jedynie do efektów spowodowanych obliczaniem wyrażeń boolowskich.

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 w sposób następujący:

```
    V1:= V:= A;
    V2:= B;
L1: IF (V1 - C) * SIGN(V2) GREATER 0
        THEN GOTO ELEMENTWYCZERPANO;
    instrukcja I;
    V2:= B;
    V1:= V:= V + V2;
    GO TO L1;
```

gdzie V jest zmienną sterowaną przez warunek cyklu, a etykieta ELEMENTYWYCZERPANO prowadzi do obliczenia następnego elementu listy cyklu lub - jeżeli rozważany element jest ostatni na liście - do następnej instrukcji programu; V1 i V2 są dodatkowymi zmiennymi roboczymi.

4.6.5. Końcowa wartość zmiennej sterowanej przez warunek cyklu. Tożsamość zmiennej sterowanej przez warunek cyklu nie jest ustalana raz na zawsze 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, jaka była bezpośrednio przed wykonaniem instrukcji skoku. Jeżeli natomiast wyjście z instrukcji I było spowodowane wyczerpaniem listy cyklu, to w przypadku, gdy ostatni element listy cyklu był (a) wyrażeniem arytmetycznym, wówczas zmienna sterowana ma jego wartość; (b) elementem postaci postępu arytmetycznego bądź elementem "podczas gdy", wówczas zmienna sterowana ma wartość równą pierwszej wartości tej zmiennej powodującej przejście do następnej instrukcji programu.

4.7.5.1. Jeżeli tekst jest parametrem aktualnym w instrukcji procedury różnej od standardowych procedur wyjścia (patrz p. 6.1. "Instrukcje wyjścia") lub w odwołaniu funkcyjnym, to ten tekst może być użyty w treści procedury jedynie jako parametr aktualny przy wywoływaniu dalszych procedur. Ostatecznie, tekstem mogą posługiwać się tylko standardowe procedury wyjścia.

4.7.5.2. Dodano następujące zdanie przy końcu punktu: Ponadto, zmienna ta musi być takiego samego typu jak specyfikowany parametr formalny.

4.7.6. Ograniczenie. Liczba parametrów formalnych procedury nie może przekraczać 127.

5.2.3.4. Segment tablic. Segment tablic może określać nie więcej niż 255 tablic.

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

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

5.4.6. Usunięto punkt 5.4.6.

10. PRZYKŁADY PROGRAMÓW W ALGOLU

W celu zwrócenia uwagi na ważne szczegóły związane z formalnym przygotowaniem problemów podajemy w tym rozdziale pełne przykłady kodowania problemów obliczeniowych w opisanej reprezentacji języka ALGOL.

Każdy z przykładów zawiera wszystkie informacje potrzebne do realizacji zadania zaprogramowanego dla maszyny ZAM 41, wyposażonej w system operacyjny SO 141. Te informacje znajdują się na załączonych tabulogramach problemów przykładowych.

Tabulogram problemu obejmuje trojakiego rodzaju informacje:

- czołówkę problemu napisaną w języku operacyjnym maszyny JOM (patrz Maszyna ZAM 41, Oprogramowanie (System 141), Tom I);
- program zakodowany w języku ALGOL, poprzedzony czołówką napisaną w języku operacyjnym ALGOLu (patrz punkt 7.1. "Język operacyjny ALGOLu") i zakończony symbolem "*";
- dane wejściowe (o ile takie są wymagane) potrzebne do realizacji programu.

Wszystkie przykłady są dokumentowane tabulogramami otrzymanymi podczas wykonywania omawianego zadania przez maszynę cyfrową ZAM 41.

Przy układaniu programów korzystano z procedur publikowanych w czasopiśmie Communications of the Association for Computing Machinery oraz w serii [Общие Вопросы Программирования - Алгоритмы].

10.1. Przykład 1

Pierwszym zaprogramowanym zadaniem jest wyznaczenie pierwiastków naturalnego stopnia z liczb zespolonych[1]. Tabulogram problemu znajduje się na stronie ALG 10-5.

Węzłowe punkty programu można opisać następująco:

- wprowadzenie za pomocą symbolicznego wejścia numer 0 liczby, która ustala ilość kolejno pierwiastkowanych liczb zespolonych;
- wprowadzanie za pomocą symbolicznego wejścia numer 0 grupy trzech liczb, z których pierwsza określa stopień pierwiastka, a druga i trzecia - odpowiednio - części rzeczywistą i urojoną pierwiastkowanej liczby zespolonej;
- wyznaczanie pierwiastków;
- wyprowadzanie za pomocą symbolicznego wyjścia numer 0 wyliczonych wartości pierwiastków.

---

1) Mostowski, A. i Stark, M., Elementy Algebry Wyższej, PWN, Warszawa 1965, str. 75-77.

Wykorzystana w programie procedura NROOT, wyznaczająca N pierwiastków równania $X^N=R+Uxi$, została opracowana przez J.R. Herndona (CACM, 1961, Nr 4 i Алгоритмы, 1966, Выпуск 3). Wartości pierwiastków są pamiętane w kolejnych miejscach tablic REALP (części rzeczywiste pierwiastków) i UNREALP (części urojone pierwiastków).

W oryginalnym sformułowaniu procedury wymienione tablice są oznaczone identyfikatorami REAL i UNREAL. Omawiana reprezentacja języka ALGOL nie dopuszcza stosowania symbolu REAL (słowo zastrzeżone) w charakterze identyfikatora. Przy przepisywaniu procedury identyfikatory REAL i UNREAL zastąpiono odpowiednio identyfikatorami REALP i UNREALP, tj. skrótami terminów real part i unreal part.

Czołówka ALGOLu zawiera dwa zdania. Ich znaczenia są opisane w punktach 7.1.1. i 7.1.3.

W czołówce JOM określono, że urządzeniem przypisanym symbolicznemu wyjściu numer 0 translatora i programu wynikowego jest drukarka wierszowa. Symbolicznemu wejściu numer 0 translatora i programu wynikowego przyporządkowuje się czytnik taśmy papierowej CTP1, a informacje na taśmie mają być perforowane w kodzie M2. Ponieważ taka definicja wchodzi w skład standardu ustalonego w JOM, nie trzeba jej podawać explicite.

Instrukcja działania maszyny użytkowej obejmuje:

- wywołanie translatora ALGOLu w celu przetłumaczenia programu w języku źródłowym na program wynikowy;
- wywołanie (tj. realizację) programu wynikowego.

Za tabulogramem problemu znajdują się tabulogramy otrzymane podczas działania maszyny. Pierwszy z nich zawiera:

- nagłówek utworzony przez system operacyjny w chwili inicjowania problemu;
- treść zdania TYTUL, przepisaną z tekstu źródłowego przez system operacyjny;
- treść programu w języku źródłowym, przepisaną przez translator ALGOLu (efekt zdania TEKST, umieszczonego w czołówce ALGOLu).

Drugi tabulogram zawiera wyniki uzyskane podczas realizacji programu wynikowego.

```
WY:O=PDW(DX,DDX1);
TYTUL: PIERWIASTKOWANIE LICZB ZESPOLONYCH;
PROBLEM: ALGOL,PROGRAM.

TEKST;
WYDAWNICTWO:O=(120,66).

BEGIN
 PROCEDURE NROOT(N,R,U)WYNIK:(REALP,UNREALP);
       VALUE N,R,U;  REAL R,U;  INTEGER N;
       ARRAY REALP,UNREALP;
 BEGIN REAL C,M,PI,S,T;  INTEGER I;
       M:=1/N;  PI:=3.14159265;
       S:=(R*R+U*U) POWER (M/2);
       T:=IF R=0 THEN SIGN(U)*PI/2 ELSE
            IF R GREATER 0 THEN ARCTAN(U/R)
             ELSE ARCTAN(U/R)+PI;
       T:=M*T;  C:=2*PI*M;
       FOR I:=1 STEP 1 UNTIL N DO
          BEGIN   REALP[I]:=S*COS(T);
                UNREALP[I]:=S*SIN(T); T:=T+C
          END I
 END PROCEDURY NROOT;
 INTEGER M,J;

 INP(O,J); COMMENT ILE LICZB PIERWIASTKOWAC;
 FOR M:=1 STEP 1 UNTIL J DO
  BEGIN
   INTEGER K;
   INP(O,K); COMMENT STOPIEN PIERWIASTKA;
   BEGIN
    INTEGER L;  REAL A,B;  COMMENT REZERWACJA
     MIEJSCA NA PIERWIASTKI;  ARRAY U,V[1:K];
    INP(O,A,B); COMMENT CZYTANIE LICZBY PIER-
                        .WIASTKOWANEJ A+B*I;
    OUT(O,'T',':3/X:BPOWER:B');
    OUT(O,'ZD',K);  OUT(O,'T',':B=:B');
    OUT(O,'Z+D',A,B);  OUT(O,'T','I:/');
    NROOT(K,A,B)WYNIK:(U,V);
    FOR L:=1 STEP 1 UNTIL K DO
     BEGIN OUT(O,'T',':/:18BX=');
           OUT(O,'Z+D.9D',U[L],V[L]);
           OUT(O,'T','I')
     END L
   END BLOKU
  END M
END PROGRAMU
*

4;
3;  8;  6;          3; -8;  6;
5;  8;  6;          5; -8;  6;
```

ALG 10-6

ZAM 41 - SO 141 BBV(A) 236/76? - ZEST 0(C) - 8.8.70

PIERWIASTKOWANIE LICZB ZESPOLONYCH

```
BEGIN
 PROCEDURE NROOT(N,R,U)WYNIK:(REALP,UNREALP);
       VALUE N,R,U;  REAL R,U;  INTEGER N;
       ARRAY REALP,UNREALP;
 BEGIN REAL C,M,PI,S,T;  INTEGER I;
       M:=1/N;  PI:=3.14159265;
       S:=(R*R+U*U) POWER (M/2);
       T:=IF R=0 THEN SIGN(U)*PI/2 ELSE
            IF R GREATER 0 THEN ARCTAN(U/R)
              ELSE ARCTAN(U/R)+PI;

LINIA 10
       T:=M*T;  C:=2*PI*M;
       FOR I:=1 STEP 1 UNTIL N DO
          BEGIN   REALP[I]:=S*COS(T);
                  UNREALP[I]:=S*SIN(T); T:=T+C
          END I
 END PROCEDURY NROOT;
 INTEGER M,J;

 INP(0,J); COMMENT ILE LICZB PIERWIASTKOWAC;
 FOR M:=1 STEP 1 UNTIL J DO

LINIA 20
  BEGIN
   INTEGER K;
   INP(0,K); COMMENT STOPIEN PIERWIASTKA;
   BEGIN
    INTEGER L;  REAL A,B;  COMMENT REZERWACJA
     MIEJSCA NA PIERWIASTKI;  ARRAY U,V[1:K];
    INP(0,A,B); COMMENT CZYTANIE LICZBY PIER-
                        WIASTKOWANEJ A+B*I;
    OUT(0,'T',':3/X:BPOWER:B');
    OUT(0,'ZD',K);  OUT(0,'T',':B=:B');

LINIA 30
    OUT(0,'Z+D',A,B);  OUT(0,'T','I:/');
    NROOT(K,A,B)WYNIK:(U,V);
    FOR L:=1 STEP 1 UNTIL K DO
     BEGIN OUT(0,'T',':/:18BX=');
           OUT(0,'Z+D.9D',U[L],V[L]);
           OUT(0,'T','I')
     END L
   END BLOKU
  END M
END PROGRAMU
*
```

```
X POWER 3 = +8+6I

                    X=+2.105061224+0.458591405I
                    X=-1.449682415+1.593740797I
                    X=-0.655378815-2.052332196I

X POWER 3 = -8+6I

                    X=+1.449682421+1.593740792I
                    X=-2.105061222+0.458591413I
                    X=+0.655378793-2.052332203I

X POWER 5 = +8+6I

                    X=+1.571785415+0.203413470I
                    X=+0.292250701+1.557714980I
                    X=-1.391164548+0.759307337I
                    X=-1.152037679-1.088437236I
                    X=+0.679166103-1.431998546I

X POWER 5 = -8+6I

                    X=+1.391164551+0.759307332I
                    X=-0.292250695+1.557714981I
                    X=-1.571785415+0.203413476I
                    X=-0.679166118-1.431998539I
                    X=+1.152037668-1.088437248I
KONIEC PROGRAMU
```

10.2. Przykład 2

Drugim zaprogramowanym zadaniem jest rozwiązanie równania

$$\sin(x) - 2x - 2 = 0$$

metodą równego podziału. Tabulogram problemu znajduje się na stronie ALG 10-10.

Zasadniczym elementem programu jest procedura BISEC opracowana przez S.Gorna (CACM, 1960, Nr 3 i Алгоритмы, 1966, выпуск 2). Procedura BISEC opisuje algorytm szukania pierwiastka funkcji F(X) w przedziale domkniętym [A,B] [1].

Warunkiem rozpoczęcia poszukiwań jest zmiana znaków wartości funkcji F(X) na końcach przedziału [A,B]. Jeżeli ten warunek nie jest spełniony, to następuje skok do instrukcji oznaczonej etykietą SIGNAL.

Pierwiastek wyznacza się iteracyjnie. Przedział jest dzielony na dwa równe podprzedziały; do dalszych obliczeń wybiera się ten podprzedział, na którego końcach wartości funkcji mają różne znaki. Opisany proces jest kontynuowany do chwili, gdy

- bezwzględna wartość funkcji stanie się nie większa od zadanej wartości EPS, lub gdy

---

[1] Demidowicz, B.P. i Maron, I.A., Metody Numeryczne, Część I, PWN, W-wa 1965, str.99-100.

- długość przedziału stanie się mniejsza od zadanej wartości EPS1.

Główne punkty programu:

- ustalenie lewego końca przedziału [A,B]. Początkowo przyjmuje się przedział [-1.3, -1];
- wielokrotne wyznaczenie pierwiastka przy różnych wartościach EPS1.

Program zawiera wszystkie parametry liczbowe niezbędne do wykonania obliczeń.

Czołówka ALGOLu składa się z jednego zdania - TEKST.

W czołówce JOM określono, że

- symbolicznemu wyjściu numer 0 translatora przypisuje się drukarkę wierszową;
- symbolicznemu wyjściu numer 0 programu wynikowego przypisuje się perforator taśmy papierowej (symbole perforowane w kodzie M2). Wyszczególnienie tego ostatniego przypisania jest właściwie zbędne, ponieważ jest ono zawarte w standardzie definicji symbolicznych wyjść.

Instrukcja działania maszyny użytkowej obejmuje:

- wywołanie translatora języka ALGOL, a następnie
- wywołanie programu wynikowego.

Na kolejnych stronach, za tabulogramem problemu, umieszczono tabulogramy otrzymane podczas pracy maszyny. Tabulogram wyników progra-

mu wynikowego uzyskano po odczytaniu taśmy perforowanej.

```
WYJSCIA TRANSLATOROW:   O=PDW(DX,DDX1);
WYJSCIA PROGRAMOW:      O=PPTP(PTP1,DM2P);
TYTUL: SZUKANIE PIERWIASTKA FUNKCJI JEDNEJ
 ZMIENNEJ F(X) W PRZEDZIALE [A,B];
PROBLEM: ALGOL, PROGRAM.

TEKST.

BEGIN
   PROCEDURE BISEC(A,B,EPS,EPS1,F,SIGNAL,X);
         VALUE A,B;  REAL A,B,EPS,EPS1,X;
         REAL PROCEDURE F;  LABEL SIGNAL;
   BEGIN REAL Y,Z;
         PROCEDURE FUN(Y);  REAL Y;
         BEGIN Y:=F(X);
               IF ABS(Y) NOTGREATER EPS
                 THEN GO TO FIN;
         END;
         X:=A;  FUN(Y);  X:=B;  FUN(Z);
         IF SIGN(Y)=SIGN(Z)
           THEN GOTO SIGNAL;
   ITER: X:=A/2+B/2;
         FUN(Y);
         IF SIGN(Y)=SIGN(Z) THEN B:=X
                            ELSE A:=X;
         IF ABS(A-B) NOTLESS EPS1
           THEN GOTO ITER;
   FIN:  END BISEC;

   REAL PROCEDURE G(X);  VALUE X;  REAL X;
         G:=SIN(X)-2*X-2;

   REAL A,B,EPS,EPS1,X;

   OUT(0,'T',':2BA:6BB:7BEPS:5BEPS1:10BX');
   OUT(0,'/');
   A:=-1.3;  B:=-1;  EPS:=1E-7;  EPS1:=1E-1;
E4:OUT(0,'/');  OUT(0,'Z-D.DDBB',A,B);
E2:BISEC(A,B,EPS,EPS1,G,E1,X);
   OUT(0,'E2B.D0E-D',EPS,EPS1);
   OUT(0,'Z7B-D.5D',X);
   IF EPS1 NOTGREATER 1E-5 THEN GO TO E3
     ELSE BEGIN EPS1:=EPS1/10; GOTO E4 END;
E1:OUT(0,'T',':18BBRAK:BPIERWIASTKA');
   A:=A-0.2;  GO TO E4;
E3:OUT(0,'/');
END
*
```

```
ZAM 41 - SO 141 BBV(A) 236/762 - ZEST 0(C)    5.8

 SZUKANIE PIERWIASTKA FUNKCJI JEDNEJ
 ZMIENNEJ F(X) W PRZEDZIALE [A,B]




BEGIN
   PROCEDURE BISEC(A,B,EPS,EPS1,F,SIGNAL,X);
         VALUE A,B;  REAL A,B,EPS,EPS1,X;
         REAL PROCEDURE F;  LABEL SIGNAL;
   BEGIN REAL Y,Z;
         PROCEDURE FUN(Y);  REAL Y;
         BEGIN Y:=F(X);
               IF ABS(Y) NOTGREATER EPS
                 THEN GO TO FIN;
         END;

LINIA 10
         X:=A;  FUN(Y);  X:=B;  FUN(Z);
         IF SIGN(Y)=SIGN(Z)
           THEN GOTO SIGNAL;
   ITER: X:=A/2+B/2;
         FUN(Y);
         IF SIGN(Y)=SIGN(Z) THEN B:=X
                            ELSE A:=X;
         IF ABS(A-B) NOTLESS EPS1
           THEN GOTO ITER;
   FIN:  END BISEC;

LINIA 20

   REAL PROCEDURE G(X);  VALUE X;  REAL X;
        G:=SIN(X)-2*X-2;

   REAL A,B,EPS,EPS1,X;

   OUT(0,'T',':2BA:6BB:7BEPS:5BEPS1:10BX');
   OUT(0,'/');
   A:=-1.3;  B:=-1;  EPS:=1E-7;  EPS1:=1E-1;
E4:OUT(0,'/');  OUT(0,'Z-D.DDBB',A,B);

LINIA 30
E2:BISEC(A,B,EPS,EPS1,G,E1,X);
   OUT(0,'E2B.DDE-D',EPS,EPS1);
   OUT(0,'Z7B-D.5D',X);
   IF EPS1 NOTGREATER 1E-5 THEN GO TO E3
     ELSE BEGIN EPS1:=EPS1/10; GOTO E4 END;
E1:OUT(0,'T',':18BBRAK:BPIERWIASTKA');
   A:=A-0.2;  GO TO E4;
E3:OUT(0,'/');
END
*
```

A	B	EPS	EPS1	X
-1.30	-1.00			BRAK PIERWIASTKA
-1.50	-1.00	.10E-6	.10E 0	-1.43750
-1.50	-1.00	.10E-6	.10E-1	-1.49219
-1.50	-1.00	.10E-6	.10E-2	-1.49902
-1.50	-1.00	.10E-6	.10E-3	-1.49872
-1.50	-1.00	.10E-6	.10E-4	-1.49870

KONIEC PROGRAMU

10.3. Przykład 3

Wróćmy jeszcze do przykładu omówionego w poprzednim punkcie. Przypuśćmy, że w celu zbadania efektywności metody równego podziału zastosowanej do rozwiązania zadania, trzeba znać liczbę iteracji wykonanych przy wyznaczaniu pierwiastka. Aby uzyskać tę dodatkową informację, należy przewidzieć w programie wykonanie odpowiednich czynności.

Język operacyjny ALGOLu zapewnia możliwość warunkowego włączania do programu wynikowego instrukcji lub deklaracji, występujących w tekście źródłowym jako komentarze (punkt 7.1.2. "Zdanie KONTROLA").

Tabulogram problemu jest umieszczony na stronie ALG 10-14. Różni się on od tabulogramu problemu przytoczonego w punkcie 10.2 w kilku miejscach.

Czołówka ALGOLu została rozszerzona - dodano zdanie KONTROLA.

Tekst źródłowy programu został powiększony o 5 struktur, odpowiadających warunkom opisanym w punkcie 7.1.2. Do programu będą włączone następujące instrukcje i deklaracja (kolej-

ność wyszczególniania odpowiada kolejności wystąpienia na tabulogramie):

- instrukcja przypisania, powodująca przypisanie zmiennej globalnej J wartości zero (pierwsza wykonywana instrukcja procedury BISEC);
- instrukcja przypisania, powodująca zwiększenie wartości zmiennej J o 1 (w treści procedury BISEC po każdym dzieleniu przedziału na pół);
- deklaracja zmiennej globalnej J typu INTEGER;
- instrukcja wyjścia tekstowego, powodująca dopisanie do nagłówka słów LICZBA ITERACJI;
- instrukcja wyjścia liczbowego, powodująca dopisywanie za wartością pierwiastka wartości zmiennej J.

Istotne zdania czołówki JOM pozostały niezmienione.

Bezpośrednio za tabulogramem problemu zamieszczono tabulogramy otrzymane w czasie realizacji problemu.

Warto podkreślić, że usunięcie zdania KONTROLA z czołówki ALGOLu spowoduje traktowanie wszystkich komentarzy występujących w programie w zwykły sposób (punkt 2.3.2. "Komentarze"). Tabulogram wyników uzyskany podczas działania programu wynikowego będzie wtedy identyczny z odpowiednim tabulogramem przytoczonym w punkcie 10.2.

```
WYJSCIA TRANSLATOROW:    O=PDW(DX,DDX1);
WYJSCIA PROGRAMOW:       O=PPTP(PTP1,DM2P);
TYTUL: SZUKANIE PIERWIASTKA FUNKCJI JEDNEJ
 ZMIENNEJ F(X) W PRZEDZIALE [A,B] -
 WARUNKOWE ZLICZANIE LICZBY ITERACJI;
PROBLEM: ALGOL, PROGRAM.

KONTROLA;
TEKST.

BEGIN
   PROCEDURE BISEC(A,B,EPS,EPS1,F,SIGNAL,X);
         VALUE A,B;  REAL A,B,EPS,EPS1,X;
         REAL PROCEDURE F;  LABEL SIGNAL;
   BEGIN REAL Y,Z;
         PROCEDURE FUN(Y);  REAL Y;
         BEGIN Y:=F(X);
               IF ABS(Y) NOTGREATER EPS
                 THEN GO TO FIN;
         END;
         COMMENT*J:=0;
         X:=A;  FUN(Y);  X:=B;  FUN(Z);
         IF SIGN(Y)=SIGN(Z)
           THEN GOTO SIGNAL;
   ITER: X:=A/2+B/2;  COMMENT*J:=J+1;
         FUN(Y);
         IF SIGN(Y)=SIGN(Z) THEN B:=X
                            ELSE A:=X;
         IF ABS(A-B) NOTLESS EPS1
           THEN GOTO ITER;
   FIN:  END BISEC;

   REAL PROCEDURE G(X);  VALUE X;  REAL X;
         G:=SIN(X)-2*X-2;

   REAL A,B,EPS,EPS1,X;  COMMENT*INTEGER J;

   OUT(0,'T',':2BA:6BB:7BEPS:5BEPS1:10BX');
   COMMENT*OUT(0,'T',':10BLICZBA:BITERACJI');
   OUT(0,'/');
   A:=-1.3;  B:=-1;  EPS:=1E-7;  EPS1:=1E-1;
E4:OUT(0,'/');  OUT(0,'Z-D.DDBB',A,B);
E2:BISEC(A,B,EPS,EPS1,G,E1,X);
   OUT(0,'E2B.DDE-D',EPS,EPS1);
   OUT(0,'Z7B-D.5D',X);
   COMMENT*OUT(0,'Z10B5D',J);
   IF EPS1 NOTGREATER 1E-5 THEN GO TO E3
     ELSE BEGIN EPS1:=EPS1/10; GOTO E4 END;
E1:OUT(0,'T',':18BBRAK:BPIERWIASTKA');
   A:=A-0.2;  GO TO E4;
E3:OUT(0,'/');
END
*
```

```
ZAM 41 - SU 141 BBV(A) 236/762 - ZEST O(C) - 8,8

 SZUKANIE PIERWIASTKA FUNKCJI JEDNEJ
 ZMIENNEJ F(X) W PRZEDZIALE [A,B] -
 WARUNKOWE ZLICZANIE LICZBY ITERACJI



BEGIN
   PROCEDURE BISEC(A,B,EPS,EPS1,F,SIGNAL,X);
         VALUE A,B;  REAL A,B,EPS,EPS1,X;
         REAL PROCEDURE F;  LABEL SIGNAL;
   BEGIN REAL Y,Z;
         PROCEDURE FUN(Y);  REAL Y;
         BEGIN Y:=F(X);
               IF ABS(Y) NOTGREATER EPS
                 THEN GO TO FIN;
         END;

LINIA 10
         COMMENT*J:=0;
         X:=A;  FUN(Y);  X:=B;  FUN(Z);
         IF SIGN(Y)=SIGN(Z)
           THEN GOTO SIGNAL;
   ITER: X:=A/2+B/2;  COMMENT*J:=J+1;
         FUN(Y);
         IF SIGN(Y)=SIGN(Z) THEN B:=X
                            ELSE A:=X;
         IF ABS(A-B) NOTLESS EPS1
           THEN GOTO ITER;

LINIA 20
   FIN:  END BISEC;

   REAL PROCEDURE G(X); VALUE X;  REAL X;
         G:=SIN(X)-2*X-2;

   REAL A,B,EPS,EPS1,X;  COMMENT*INTEGER J;

   OUT(0,'T',':2BA:6BB:7BEPS:5BEPS1:10BX');
   COMMENT*OUT(0,'T',':10BLICZBA:BITERACJI');
   OUT(0,'/');

LINIA 30
   A:=-1.3;  B:=-1;  EPS:=1E-7;  EPS1:=1E-1;
E4:OUT(0,'/');  OUT(0,'Z-D.DDBB',A,B);
E2:BISEC(A,B,EPS,EPS1,G,E1,X);
   OUT(0,'E2B.DDE-D',EPS,EPS1);
   OUT(0,'Z7B-D.5D',X);
   COMMENT*OUT(0,'Z10B5D',J);
   IF EPS1 NOTGREATER 1E-5 THEN GO TO E3
     ELSE BEGIN EPS1:=EPS1/10; GOTO E4 END;
E1:OUT(0,'T',':18BBRAK:BPIERWIASTKA');
   A:=A-0.2;  GO TO E4;

LINIA 40
E3:OUT(0,'/');
END
*
```

A	B	EPS	EPS1	X	LICZBA ITERACJI
-1.30	-1.00			BRAK PIERWIASTKA	
-1.50	-1.00	.10E-6	.10E 0	-1.43750	3
-1.50	-1.00	.10E-6	.10E-1	-1.49219	6
-1.50	-1.00	.10E-6	.10E-2	-1.49902	9
-1.50	-1.00	.10E-6	.10E-3	-1.49872	13
-1.50	-1.00	.10E-6	.10E-4	-1.49870	16

KONIEC PROGRAMU

10.4. Przykład 4

W przypadku konieczności wielokrotnej realizacji tych samych programów w ciągu dłuższego okresu czasu, problemy należy tak przygotowywać, aby unikać wielokrotnego tłumaczenia tekstu źródłowego na program wynikowy.

Są dwa sposoby postępowania, pozwalające wyprowadzić z pamięci maszyny program wynikowy (utworzony przez translator języka ALGOL) uzupełniony opisującymi go informacjami - w postaci nadającej się do ponownego wprowadzenia.

Metoda pierwsza sprowadza się do wykorzystania możliwości przewidzianych w języku operacyjnym ALGOLu (punkt 7.1.4. "Zdanie WYPROWADZ PROGRAM WYNIKOWY"). Metoda druga polega na wykorzystaniu odpowiednich translatorów wchodzących w skład systemu oprogramowania SO 141 (patrz Maszyna ZAM 41, System Oprogramowania SO 141, Opisy Funkcjonalne,(materiały aktualizowane), BIN(1)). Zastosowanie jej do programów kodowanych w języku ALGOL wyjaśniono za pomocą przykładu przytoczonego w punkcie 10.5.

Metodę pierwszą ilustrujemy przykładem programu mnożenia macierzy. Tabulogram problemu zamieszczono na stronach ALG 10-21 i ALG 10-22.

Główne punkty programu:

- wprowadzenie za pomocą wejścia symbolicznego o numerze 0 wymiarów macierzy A (kolejno: liczba wierszy, liczba kolumn) i wymiarów macierzy B;
- wprowadzenie za pomocą wejścia symbolicznego o numerze 0 elementów macierzy A i B;
- wyprowadzenie za pomocą wyjścia symbolicznego o numerze 0 elementów macierzy A i B;
- obliczenie iloczynu C = A × B;
- wyprowadzenie za pomocą wyjścia symbolicznego o numerze 0 elementów macierzy C.

Czołówka ALGOLu składa się z dwóch zdań: zdania WYPROWADZ PROGRAM WYNIKOWY i zdania WYDAWNICTWO

W czołówce problemu podano, że:

- symbolicznemu wyjściu numer 0 translatora i programu wynikowego przypisuje się drukarkę wierszową;
- symbolicznemu wyjściu numer 1 translatora przypisuje się perforator taśmy papierowej (symbole perforowane w kodzie M2). Zdefiniowanie symbolicznego wyjścia o numerze 1 dla translatora jest konieczne ze względu na umieszczone w czołówce ALGOLu zdanie WYPROWADZ PROGRAM WYNIKOWY.

Instrukcja działania maszyny użytkowej obejmuje:

- wywołanie translatora języka ALGOL, a następnie
- wywołanie programu wynikowego.

Podczas realizacji programu wynikowego ma być wyliczony iloczyn:

$$\begin{pmatrix} 1 & -1 & 1 \\ 2 & 1 & 0 \\ 5 & 7 & 1 \\ 1 & 4 & 3 \end{pmatrix} * \begin{pmatrix} 5 & -1 & 2 & 1 & -1 \\ 2 & 2 & 0 & 1 & 1 \\ 1 & -1 & 5 & 4 & 4 \end{pmatrix}$$

Dane wejściowe odpowiadające elementom macierzy A i B występują na tabulogramie problemu w takiej kolejności, jaką opisano w punkcie 6.1.3.

Podczas realizacji problemu otrzymano tabulogram z wynikami na drukarce wierszowej (jest on umieszczony bezpośrednio za tabulogramem problemu) oraz taśmę papierową z wyperforowanym programem wynikowym i opisującymi go parametrami. Taśma została wyprodukowana po utworzeniu programu wynikowego przez translator. Tabulogram zawartości taśmy papierowej jest przedstawiony w punkcie 10.4.1.[1]. Pierwszą informacją na taśmie jest czołówka napisana w języku operacyjnym ALGOLu. Składa się ona ze zdania PROGRAM WYNIKOWY. Ostatnią informacją jest makrorozkaz FINISH.

Przy kolejnym kompletowaniu problemu mnożenia macierzy, wyprodukowaną taśmę włącza się w miejsce programu źródłowego wraz z czołówką ALGOLu i symbolem kończącym "*". Czołówkę problemu przygotowuje się tak jak dla programów pisanych w ALGOLu.

W punkcie 10.4.2. zamieszczono odpowiedni tabulogram czołówki problemu oraz danych wejś-

1) Program wynikowy jest napisany w języku makrorozkazów zdefiniowanym w książce B. Randell i L.J. Russell - ALGOL 60 Implementation, AP, London and New York 1964.

ciowych problemu mnożenia macierzy. Do pamięci maszyny wprowadza się kolejno:

- czołówkę problemu,
- program wynikowy wraz z informacjami go opisującymi (punkt 10.4.1.),
- dane wejściowe.

W instrukcji działania maszyny użytkowej wymieniono dwukrotnie żądanie wywołania programu wynikowego. Tak więc program wynikowy zostanie zrealizowany dla dwóch wariantów danych wejściowych.

Dalsze tabulogramy zamieszczone w punkcie 10.4.2. otrzymano podczas realizacji problemu przez maszynę cyfrową ZAM 41.

```
WY:0=PDW/DX,DDX1/;        WYT:1=PPTP/PTP1,DM2P/;
TYTUL: MNOŻENIE MACIERZY - TEKST W ALGOLU;
PROBLEM: ALGOL, PROGRAM.

WYPROWADZ PROGRAM WYNIKOWY;
WYDAWNICTWO:O = /120,66/.

BEGIN
  PROCEDURE PISZ/A,M,N,NAPIS,WZORZEC/;
        VALUE M,N;    INTEGER M,N; ARRAY  A;
        STRING NAPIS,WZORZEC;
  BEGIN INTEGER I, J;
        OUT/O,* T,* : 3/*,NAPIS , ':/'/;
        FOR I:=1 STEP 1 UNTIL M DO
          BEGIN OUT/O, '/' /;
                FOR J:=1 STEP 1 UNTIL N DO
                  OUT /O, WZORZEC , A[I,J]/;

          END FOR I
  END PROCEDURE PISZ;

  PROCEDURE ILOCZYN /A,B,M,L,N,C/;
        VALUE M,L,N;  INTEGER M,L,N;
        ARRAY A,B,C;
  BEGIN REAL R; INTEGER I,J,K;
        FOR I:=1 STEP  1 UNTIL M DO
         FOR J:=1 STEP  1 UNTIL N DO
          BEGIN R:=0;
                FOR K:=1 STEP  1 UNTIL L DO
                 R:=R+A [I,K] * B[K,J,];
                C[I,J]:=R
          END FOR I,J
  END PROCEDURE ILOCZYN MACIERZY;

  INTEGER P,Q,R,S;

  INP/O,P,Q,R,S/;
  BEGIN
    ARRAY A[1:P,1:Q],B[1:R,1:S], C[1:P,1:S];

    INP/O,A,B/;
    PISZ/A,P,Q,'MACIERZ:BA','Z-D.D5B'/;
    PISZ/B,R,S,'MACIERZ:BB','Z-D.D5B'/;
    IF Q NOTEQUAL R THEN GOTO E;
    ILOCZYN /A,B,P,R,S,C/;
    PISZ/C,P,S,'ILOCZYN:BMACIERZY:BA*B',
             'Z-2D.2D3B'/;
    OUT/O,'3/'/;  GOTO F ;
  E:OUT/O,'T', ':/BLEDNE:BWYMIARY:BMACIERZY'/;
F:END
END *
```

```
WYMIARY MACIERZY A
WIERSZY: 4;      KOLUMN: 3;

WYMIARY MACIERZY B
WIERSZY: 3;      KOLUMN: 5;

ELEMENTY MACIERZY A:
 1, 2, 5, 1,
-1, 1, 7, 4,
 1, 0, 1, 3;

ELEMENTY MACIERZY B:
 5, 2, 1,
-1, 2,-1,
   0, 5,
      1, 4,
      1, 4;
```

ZAM 41 - SO 141 BBW(A) 320/002 - ZEST 0(C) - 13.8

MNOZENIE MACIERZY - TEKST W ALGOLU

MACIERZ A

1.0	-1.0	1.0
2.0	1.0	0.0
5.0	7.0	1.0
1.0	4.0	3.0

MACIERZ B

5.0	-1.0	2.0	1.0	-1.0
2.0	2.0	0.0	1.0	1.0
1.0	-1.0	5.0	4.0	4.0

ILOCZYN MACIERZY A*B

4.00	-4.00	7.00	4.00	2.00
12.00	0.00	4.00	3.00	-1.00
40.00	8.00	15.00	16.00	6.00
16.00	4.00	17.00	17.00	15.00

KONIEC PROGRAMU

10.4.1. Tabulogram zawartości taśmy perforowanej

```
PROGRAM WYNIKOWY.
X00100104
X25403360
X47475062
X06041000
X00001623
X00002262
X00000000
X00000000
X00000000
X00000000
X00000002
X00000000
X00000000
X00000000
X00000000
X00000141
X00000551
X00000066
X00014275
X00014275
X00074102
X00042514
X00042514
X00042514
X00042514
X00042514
X00042514
X00042514
X00000000
X00000000
X00000000
X00000000
X00000000
X00000000
X00000000
X00000000
X00013772
X00014134
X00000000
X00000000
X00000000
X00000000
X00000000
X00000000
X00000000
X00000000
X00000000
X00000000
```

X00011547
X00000000
X00000000
X00000000
X00000000
X00000000
X00000000
X00000003
X00000006
X00000006
X00000006
X00000013
X00000041
X00000060
X00000074
X00000113
X00000137
X00000137
X00000141
X00000141
X00000143
X00000143
X00000143
X00000151
X00000170
X00000207
X00000212
X00000231
X00000247
X00000253
X00000255
X00000260
X00000260
X00000260
X00000260
X00000301
X00000301
X00000326
X00000326
X00000343
X00000370
X00000415
X00000424
X00000444
X00000455
X00000476
X00000516
X00000546
X00000547

*

```
 0  CBL
 1  UJ      (  360)
 2  BE      ( 1,   4)
 3  UJ      (  176)
 4  PE      ( 2,  12),  5
 6  CAR
 7  CSI
 8  CSI
 9  CST
10  CST
11  UJ      (   30)
12  TIC     '0000000040000377'
15  STRING  'T'
16  STRING  ':3/'
18  STRING  ':/'
20  PST     (   18)
22  PF      ( 2,   9)
24  PST.    (  162
26  PST     (   15)
28  PIC     (   13)
30  CFOUT   (   10),  5
32  REJECT
33  UJ      (   36)
34  TIA     ( 2,  13)
35  LINK
36  CFZ     (   46)
37  FORS1
38  TIC1
39  LINK
40  FORS2
41  TIC1
42  LINK
43  TIR     ( 2,   5)
44  LINK
45  FSE     (   96)
46  FBE     ( 3,   4),(   34)
48  UJ      (   57)
49  TIC     '0000000040000377'
52  STRING  '/'
53  PST     (   52)
55  PIC     (   50)
57  CFOUT   (   10),  2
59. REJECT
60  UJ      (   63)
61  TIA     ( 2,  14)
62  LINK
63  CFZ     (   73)
64  FORS1
65  TIC1
66  LINK
67  FORS2
68  TIC1
```

```
 69  LINK
 70  TIR     ( 2,   7)
 71  LINK
 72  FSE     (   95)
 73  FBE     ( 4,   4),(  61)
 75  UJ      (   91)
 76  TIC     '0000000040000377'
 79  BE      ( 5,   0)
 80  TFA     ( 2,   3)
 81  TIR     ( 2,  13)
 82  TIR     ( 2,  14)
 83  INDA
 84  EIS
 85  PSR     (   79)
 87  PF      ( 2,  11)
 89  PIC     (   77)
 91  CFOUT   (   10),  3
 93  REJECT
 94  FR
 95  FR
 96  RETURN
 97  PE      ( 2,  16),  6
 99  CAR
100  CAR
101  CSI
102  CSI
103  CSI
104  CAR
105  UJ      (  108)
106  TIA     ( 2,  16)
107  LINK
108  CFZ     (  118)
109  FORS1
110  TIC1
111  LINK
112  FORS2
113  TIC1
114  LINK
115  TIR     ( 2,   7)
116  LINK
117  FSE     (  175)
118  FBE     ( 3,   4),(  106)
120  UJ      (  123)
121  TIA     ( 2,  17)
122  LINK
123  CFZ     (  133)
124  FORS1
125  TIC1
126  LINK
127  FORS2
128  TIC1
129  LINK
```

```
130  TIR     ( 2,  11)
131  LINK
132  FSE     (  174)
133  FBE     ( 4,   4),(  121)
135  TRA     ( 2,  15)
136  TICO
137  ST
138  UJ      (  141)
139  TIA     ( 2,  18)
140  LINK
141  CFZ     (  151)
142  FORS1
143  TIC1
144  LINK
145  FORS2
146  TIC1
147  LINK
148  TIR     ( 2,   9)
149  LINK
150  FSE     (  167)
151  FBE     ( 5,   4),(  139)
153  TRA     ( 2,  15)
154  TRR     ( 2,  15)
155  TFA     ( 2,   3)
156  TIR     ( 2,  16)
157  TIR     ( 2,  18)
158  INDR
159  TFA     ( 2,   5)
160  TIR     ( 2,  18)
161  TIR     ( 2,  17)
162  INDR
163  *
164  +
165  ST
166  FR
167  TFA     ( 2,  13)
168  TIR     ( 2,  16)
169  TIR     ( 2,  17)
170  INDA
171  TRR     ( 2,  15)
172  ST
173  FR
174  FR
175  RETURN
176  UJ      (  190)
177  TIC     '000000004000037'
180  PI      ( 1,   6)
182  PI      ( 1,   5)
184  PI      ( 1,   4)
186  PI      ( 1,   3)
188  PIC     (  178)
190  CFINP   (    9),  5
```

```
192  REJECT
193  CBL
194  UJ      ( 359)
195  BE      ( 2,   3)
196  TIC1
197  TIR     ( 1,   3)
198  TIC1
199  TIR     ( 1,   4)
200  MSF     ( 2,   3),  1
202  TIC1
203  TIR     ( 1,   5)
204  TIC1
205  TIR     ( 1,   6)
206  MSF     ( 2,   4),  1
208  TIC1
209  TIR     ( 1,   3)
210  TIC1
211  TIR     ( 1,   6)
212  MSF     ( 2,   5),  1
214  UJ      ( 224)
215  TIC     '0000000040000377'
218  PRA     ( 2,   4)
220  PRA     ( 2,   3)
222  PIC     ( 216)
224  CFINP   (    9),  3
226  REJECT
227  UJ      ( 245)
228  STRING  'MACIERZ:BA'
232  STRING  'Z-D.D5B'
235  PST     ( 232)
237  PST     ( 228)
239  PI      ( 1,   4)
241  PI      ( 1,   3)
243  PRA     ( 2,   3)
245  CF      (    4),  5
247  REJECT
248  UJ      ( 266)
249  STRING  'MACIERZ:BB'
253  STRING  'Z-D.D5B'
256  PST     ( 253)
258  PST     ( 249)
260  PI      ( 1,   6)
262  PI      ( 1,   5)
264  PRA     ( 2,   4)
266  CF      (    4),  5
268  REJECT
269  TIR     ( 1,   4)
270  TIR     ( 1,   5)
271  NOTEQ
272  IFJ     ( 276)
273  TL      ( 334),  2
275  GTA
```

```
276  UJ       (  289)
277  PRA      ( 2,   5)
279  PI       ( 1,   6)
281  PI       ( 1,   5)
283  PI       ( 1,   3)
285  PRA      ( 2,   4)
287  PRA      ( 2,   3)
289  CF       (   97),  6
291  REJECT
292  UJ       (  315)
293  STRING   'ILOCZYN:BMACIERZY:BA*B'
301  STRING   'Z-2D,2D3B'
305  PST      (  301)
307  PST      (  293)
309  PI       ( 1,   6)
311  PI       ( 1,   3)
313  PRA      ( 2,   5)
315  CF       (    4),  5
317  REJECT
318  UJ       (  328)
319  TIC      '0000000040000377'
322  STRING   '3/'
324  PST      (  322)
326  PIC      (  320)
328  CFOUT    (   10),  2
330  REJECT
331  TL       (  358),  2
333  GTA
334  UJ       (  3552
335  TIC      '0000000040000377'
338  STRING   'T'
339  STRING   ':/BLEDNE:BWYMIARY:BMACIERZY'
349  PST      (  339)
351  PST      (  338)
353  PIC      (  336)
355  CFOUT    (   10),  3
357  REJECT
358  RETURN
359  RETURN
360  FINISH
```

10.4.2. Wprowadzanie programu wynikowego - tabulogramy uzupełniające

```
WY:0=PDW(DX,DDX1);
TYTUL:MNOZENIE MACIERZY - TEKST PROGRAMU
W JEZYKU POSREDNIM;
PROBLEM:ALGOL,PROGRAM,PROGRAM.
```

DANE LICZBOWE DLA PIERWSZEGO WYWOLANIA PROGRAMU

```
WYMIARY MACIERZY A
WIERSZY:   5;        KOLUMN:   1;

WYMIARY MACIERZY B:
           1;                  5;

A11= 2,  A21= 5,  A31= 1,  A41=-1,  A51= 4;

B11= 3,  B12= 4,  B13= 5,  B14= 0,  B15= 1;
```

DANE LICZBOWE DLA DRUGIEGO WYWOLANIA PROGRAMU:

```
3;2;
2;3;

2, 3,5,
1,-3,2;

 1,2,
-1,4,
 1,3;
```

ZAM 41 - SO 141 BBV(A) 236/762 - ZEST 0(C) - 8.8

MNOŻENIE MACIERZY - TEKST PROGRAMU
W JĘZYKU POŚREDNIM

MACIERZ A

2.0
5.0
1.0
-1.0
4.0

MACIERZ B

3.0	4.0	5.0	0.0	1.0

ILOCZYN MACIERZY A*B

6.00	8.00	10.00	0.00	2.00
15.00	20.00	25.00	0.00	5.00
3.00	4.00	5.00	0.00	1.00
-3.00	-4.00	-5.00	0.00	-1.00
12.00	16.00	20.00	0.00	4.00

KONIEC PROGRAMU

```
 MACIERZ A

  2.0       1.0
  3.0      -3.0
  5.0       2.0

 MACIERZ B

  1.0      -1.0       1.0
  2.0       4.0       3.0

 ILOCZYN MACIERZY A*B

  4.00      2.00      5.00
 -3.00    -15.00     -6.00
  9.00      3.00     11.00

KONIEC PROGRAMU
```

10.5. Przykład 5

Podany w poprzednim punkcie sposób postępowania, pozwalający wyprowadzić z pamięci maszyny program wynikowy wraz z opisującymi go parametrami odnosi się wyłącznie do programów wynikowych utworzonych przez translator języka ALGOL. Teraz przedstawimy ogólną metodę wyprowadzania programu wynikowego (i opisujących go parametrów) skompletowanego w pamięci maszyny przez translator dowolnego języka, a więc w szczególności przez translator ALGOLu.

Metoda ta polega na zastosowaniu do tego celu programów wyjścia i wejścia binarnego WYBIN i WEBIN wchodzących w skład systemu oprogramowania SO 141 (patrz Maszyna ZAM 41, System Oprogramowania SO 141, Opisy Funkcjonalne, (materiały aktualizowane), BIN(1)). Programy WYBIN i WEBIN zwane są dalej translatorami, ponieważ funkcjonują one w systemie na prawach translatorów.

W celu wyprowadzenia taśmy binarnej programu wynikowego (wraz z opisującymi go parametrami), utworzonego w pamięci maszyny przez translator ALGOLu, należy tak zdefiniować problem, aby wywołanie translatora WYBIN nastąpiło bezpośrednio po zakończeniu tłumaczenia programu źródłowego.

Translator WYBIN produkuje taśmę binarną za pomocą symbolicznych wyjść o numerach 1 i 2. Wymaga się, aby w obu tych definicjach wystąpiło to samo urządzenie oraz, aby dekoder wskazany w definicji symbolicznego wyjścia o numerze 2 był dekoderem jednostkowym -DJED.

Problem ilustrujący zastosowanie translatora WYBIN sformułowano opierając się na programie mnożenia macierzy omówionym w punkcie 10.4.

Tabulogram problemu (bez tabulogramu danych wejściowych) zamieszczono na str. ALG 10-37.

W czołówce problemu określono, że

- symbolicznemu wyjściu translatorów o numerze 1 przypisuje się perforator taśmy papierowej PTP1, symbole są perforowane w kodzie M2;
- symbolicznemu wyjściu translatorów o numerze 2 przypisuje się perforator taśmy papierowej PTP1, symbole są perforowane w kodzie jednostkowym (patrz Maszyna ZAM 41, System Oprogramowania SO 141, Opisy Funkcjonalne,(materiały aktualizowane), DOD(1));
- symbolicznemu wyjściu numer 0 translatorów i programu wynikowego przypisuje się drukarkę wierszową.

Instrukcja działania maszyny użytkowej zawiera:

- wywołanie translatora ALGOLu,
- wywołanie translatora wyjścia binarnego WYBIN,
- wywołanie programu wynikowego.

W tym przypadku z wyjścia symbolicznego o numerze 0 korzysta się kolejno podczas działania translatorów ALGOL i WYBIN w celu sygnalizacji wykrytych błędów, a następnie podczas realizacji programu wynikowego - w celu wyprowadzania otrzymanych wyników. Natomiast z wyjść symbolicznych o numerach 1 i 2 korzysta jedynie translator WYBIN.

Wyprodukowana taśma binarna zawiera wszystkie informacje przekazywane Interpreterowi ALGOLu i systemowi operacyjnemu przez translator ALGOLu. Można ją wprowadzać do pamięci maszyny za pomocą translatora WEBIN.

Translator WEBIN wprowadza taśmę binarną (przygotowaną przez WYBIN) za pomocą symbolicznego wyjścia o numerze 1. Wymaga się, aby dekoder wskazany w definicji tego wejścia był dekoderem jednostkowym -DJED. W przypadku wykrycia nieprawidłowych sytuacji podczas wprowadzania, WEBIN sygnalizuje błędy - wypisując odpowiednie teksty na monitorze (patrz Maszyna ZAM 41, System Oprogramowania SO 141, Opisy Funkcjonalne, (materiały aktualizowane), BIN(1)).

Czołówka problemu, w którym będzie wprowadzana zamiast programu w ALGOLu jego taśma binarna, ma postać (porównaj czołówkę problemu przytoczoną w punkcie 10.4.2.) następującą:

```
WEJSCIA TRANSLATOROW: 1=PCTP(CTP1,DJED);
WYJSCIA: 0=PDW(DX,DDX1);
PROBLEM: WEBIN, PROGRAM, PROGRAM.
```

```
WYT:1=PPTP(PTP1,DM2P),     2=PPTP(PTP1,DJED);
WY: 0=PDW(DX,DDX1);
TYTUL: WYPROWADZENIE INFORMACJI WYNIKOWYCH
 TRANSLATORA ALGOL W FORMIE TASMY BINARNEJ;
PROBLEM: ALGOL, WYBIN, PROGRAM.

WYDAWNICTWO: 0=(120,66).

BEGIN
  PROCEDURE PISZ(A,M,N,NAPIS,WZORZEC);
        VALUE M,N;  INTEGER M,N;  ARRAY A;
        STRING NAPIS,WZORZEC;
  BEGIN INTEGER I,J;
        OUT(0,'T',':3/', NAPIS ,':/');
        FOR I:=1 STEP 1 UNTIL M DO
          BEGIN OUT(0,'/');
                FOR J:=1 STEP 1 UNTIL N DO
                  OUT(0, WZORZEC ,A[I,J]);
          END FOR I
  END PROCEDURY PISZ;

  PROCEDURE ILOCZYN(A,B,M,L,N,C);
        VALUE M,L,N;  INTEGER M,L,N;
        ARRAY A,B,C;
  BEGIN REAL R; INTEGER I,J,K;
        FOR I:=1 STEP 1 UNTIL M DO
          FOR J:=1 STEP 1 UNTIL N DO
            BEGIN R:=0;
                  FOR K:=1 STEP 1 UNTIL L DO
                    R:=R+A[I,K]*B[K,J];
                  C[I,J]:=R
            END FOR I,J
  END PROCEDURY ILOCZYN MACIERZY;

  INTEGER P,Q,R,S;

  INP(0,P,Q,R,S);
  BEGIN
    ARRAY A[1:P,1:Q],B[1:R,1:S],C[1:P,1:S];

    INP(0,A,B);
    PISZ(A,P,Q, 'MACIERZ:BA' , 'Z-D.D5B');
    PISZ(B,R,S, 'MACIERZ:BB' , 'Z-D.D5B');
    IF Q NOTEQUAL R THEN GOTO E;
    ILOCZYN(A,B,P,R,S,C);
    PISZ(C,P,S, 'ILOCZYN:BMACIERZY:BA*B',
                'Z-2D.2D3B');
    OUT(0,'3/'); GOTO F;
  E:OUT(0,'T',':/BLEDNE:BWYMIARY:BMACIERZY');
F:END
END *
```

DODATEK A
SŁOWA ZASTRZEŻONE I ICH ZNACZENIE W JĘZYKU POLSKIM

Słowo zastrzeżone	Znaczenie	Słowo zastrzeżone	Znaczenie
AND	i	LABEL	etykieta
ARRAY	tablica	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
INTERPRETACJA SYMBOLI PRZEZ FUNKCJĘ INCHAR[1/]

symbol	liczba	symbol	liczba	symbol	liczba	symbol	liczba
NUL (NU)*	64	< (LT)	80	L (XL)	96	0 (Z0)	112
SP (SP)*	65	> (GT)	81	M (XM)	97	1 (Z1)	113
. (FP)	66	? (QS)	82	N (XN)	98	2 (Z2)	114
((LP)	67	ESC (ES)*	83	O (XO)	99	3 (Z3)	115
+ (PL)	68	SO (SO)*	84	P (XP)	100	4 (Z4)	116
* (AS)	69	A (XA)	85	Q (XQ)	101	5 (Z5)	117
) (RP)	70	B (XB)	86	R (XR)	102	6 (Z6)	118
; (SE)	71	C (XC)	87	S (XS)	103	7 (Z7)	119
- (MI)	72	D (XD)	88	T (XT)	104	8 (Z8)	120
/ (SL)	73	E (XE)	89	U (XU)	105	9 (Z9)	121
, (CM)	74	F (XF)	90	V (XV)	106	LF (LF)*	122
: (CL)	75	G (XG)	91	W (XW)	107	◊ (N6)	123
' (AP)	76	H (XH)	92	X (XX)	108	FF (FF)*	124
= (EQ)	77	I (XI)	93	Y (XY)	109	CS (CS)*	125
[(LB)	78	J (XJ)	94	Z (XZ)	110	CR (CR)*	126
] (RB)	79	K (XK)	95	SI (SI)*	111	DEL (DE)*	127

[1/] Symbole opatrzone * nie reprezentują same siebie - patrz norma ISO R 646.

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 symbólu "END"
205	wystąpienie po symbolu ")" liczby albo jednego z symboli "COMMENT" lub ","
206	niepoprawny zapis liczby lub niewłaściwie użyty separator "."
207	niepoprawny zapis ogranicznika na liście argumentów procedury
208	brak zakończenia programu symbolem "*" albo nieprawidłowa struktura blokowa

Podczas 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

Uwaga. Jeżeli po przeczytaniu taśmy z programem maszyna czyta (lub chce czytać) dalej taśmę, oznacza to, że program ma nieprawidłową strukturę, tj. liczbie użytych symboli "BEGIN" nie odpowiada taka sama liczba symboli "END" lub liczba użytych symboli "'" jest nieparzysta.

DODATEK D
BŁĘDY I PRZEPEŁNIENIA SYGNALIZOWANE W DRUGIM PRZEBIEGU

Nr błędu	Opis błędu
100	brak ogranicznika (lub ogranicznik napisany niepoprawnie) między dwoma identyfikatorami, liczbami lub wartościami logicznymi albo 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 w zbiorze specyfikacji
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;

Nr błędu	Opis błędu
108	ogranicznika "END" użyto jako symbolu kończącego 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; przykłady: X LESS Y+Z LESS 3 FOR J:= J+100 WHILE J GREATER 750 DO A[I,J]:I+J
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 tablicy
118[1)]	użycie identyfikatora niezgodne z jego deklaracją
119[1)]	identyfikator użyty niejednoznacznie; przykład: IF U NOTEQUAL U[I]

[1)] W pewnych przypadkach są wypisywane numery dwóch wierszy; pierwszy z nich wskazuje koniec aktualnie przetwarzanego bloku, drugi - wiersz, w którym występuje źle użyty identyfikator.

Nr błędu	Opis błędu
120	niepoprawny kontekst zawierający ":="
121	niepoprawne wyrażenie występujące w deklaracji tablicy
122	niepoprawne wyrażenie sterujące na liście przełączeń
123	operator arytmetyczny poprzedzono niewłaściwym symbolem; przykłady: I:=/R; OUT (0, 'E', A + -12.5);
124	niepoprawny kontekst zawierający separator ":"
125	operator relacji użyty w wyrażeniu arytmetycznym
126	niepoprawna konstrukcja wyrażenia zawierającego warunek
127	ogranicznika "THEN" nie poprzedza wyrażenie boolowskie albo wyrażenie to nie jest poprawne
128	niepoprawna budowa instrukcji warunkowej albo wyrażenia warunkowego
129	ogranicznik "GOTO" występuje w niewłaściwym kontekście
130	niepoprawna budowa wyrażenia boolowskiego albo użycie operatora logicznego w wyrażeniu arytmetycznym
131	niepełna struktura poprzedzająca ogranicznik "FOR"; przykłady: X:=X+1 FOR I:=0 STEP 1 UNTIL X DO A [I] :=0;

Nr błędu	Opis błędu
	FOR I:=1, 10 FOR J:=0 STEP 1 UNTIL 10 DO A [I,J]:=0;
132	separator "STEP", "UNTIL" albo "WHILE" występuje w niewłaściwym kontekście albo brak separatora "UNTIL"
133	błąd w kontekście zawierającym "(", na przykład brak operatora
134	")" występuje w niewłaściwym kontekście
136	niepoprawnie zbudowane wyrażenie indeksowe
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 nie będącej odwołaniem funkcyjnym
143	nawias "[" nie jest poprzedzony identyfikatorem lub zmienna indeksowana bądź przełączenie występuje w błędnym kontekście
144	w parach ograniczeń tablicy własnej użyto identyfikatora albo niewłaściwego symbolu
145	ogranicznik niepoprawnie użyty w parach ograniczeń tablicy własnej
146	za duża liczba elementów tablicy własnej

Nr błędu	Opis błędu
147	separator ":" lub "," występuje w niepoprawnym kontekście
148	separator ":" lub "," błędnie użyty w deklaracji tablicy
149	separator ":" występuje w wyrażeniu indeksowym
150[1)]	liczba indeksów zmiennej indeksowanej nie odpowiada liczbie indeksów w deklaracji tablicy 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 tablicy
153	ponowna deklaracja tego samego identyfikatora
154	niewłaściwa budowa bloku
155[1)]	użycie identyfikatora niezgodne z jego deklaracją
156[1)]	użycie identyfikatora procedury funkcji po lewej stronie symbolu ":=" w instrukcji przypisania poza treścią tej procedury
157[1)]	użycie identyfikatora procedury nie będącej funkcją w instrukcji przypisania

1) patrz odnośnik na str. D-2

Nr błędu	Opis błędu
158[1)]	użycie identyfikatora niezgodne z jego deklaracją
159[1)]	w niewłaściwy sposób wywołana procedura bez parametrów, nie będąca funkcją
160[1)]	wywołanie procedury niefunkcyjnej
161[1)]	niewłaściwe wywołanie procedury bez parametrów
162	brak specyfikacji jednego z parametrów formalnych
163	na liście parametrów wywoływanych przez wartość występuje parametr nie mający wartości
164	ponowne umieszczenie parametru formalnego w specyfikacjach albo specyfikowanie identyfikatora nie będącego parametrem formalnym
165	ponowne umieszczenie 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
169	wystąpienie nawiasów " (" ") " w błędnym kontekście

[1)] patrz odnośnik na str. D-2

Nr błędu	Opis błędu
170	wyrażenia dla ograniczeń indeksów zależą od zmiennej lokalnej bądź funkcji lokalnej w bloku, w którym obowiązuje deklaracja tablicy
171	jak 156
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; uwaga: instrukcja występująca za warunkiem cyklu jest zawsze traktowana jako blok.

Podczas 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 lub błąd wynikający z poprzednich błędów
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[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 odpowiedniego 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
9	niewłaściwy typ zmiennej występującej po znaku "-" w wyrażeniu arytmetycznym

1) pierwszy parametr za tekstem LINIA określa wiersz, w którym rozpoczyna się deklaracja procedury, drugi parametr za tekstem LINIA określa miejsce wywołania procedury.

Nr błędu	Opis błędu
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 podstawy przy potęgowaniu
16	niewłaściwy typ wykładnika przy potęgowaniu
17	nieokreślony wynik potęgowania: podstawa ujemna, wykładnik rzeczywisty
18	niewłaściwy typ argumentu operacji NOT
19[1)]	niewłaściwy parametr aktualny
20[1)]	niewłaściwy parametr aktualny; na przykład trzeci parametr procedury rozmieszczenia nie jest tekstem
21[1)]	niewłaściwy parametr aktualny (nie odpowiada specyfikacji REAL)
22[1)]	niewłaściwy parametr aktualny (nie odpowiada specyfikacji BOOLEAN)
23[1)]	niewłaściwy parametr aktualny (nie odpowiada specyfikacji LABEL)

1) patrz odnośnik na stronie E-1

Nr błędu	Opis błędu
24[1]	niewłaściwy parametr aktualny (nie odpowiada specyfikacji INTEGER lub REAL)
25[1]	typ parametru aktualnego niezgodny ze specyfikacją REAL parametru formalnego, który w treści procedury występuje jako element listy lewych stron w instrukcji przypisania
26[1]	typ parametru aktualnego niezgodny ze specyfikacją INTEGER parametru formalnego, który w treści procedury występuje jako element listy lewych stron w instrukcji przypisania
27[1]	typ parametru aktualnego niezgodny ze specyfikacją REAL parametru formalnego, umieszczonego na liście parametrów wywoływanych przez wartość
28[1]	typ parametru aktualnego niezgodny ze specyfikacją INTEGER parametru formalnego, umieszczonego na liście parametrów wywoływanych przez wartość
29[1]	typ parametru aktualnego niezgodny ze specyfikacją BOOLEAN parametru formalnego, umieszczonego na liście parametrów wywoływanych przez wartość
30[1]	typ parametru aktualnego niezgodny ze specyfikacją LABEL parametru for-

[1] patrz odnośnik na stronie E-1

Nr błędu	Opis błędu
	malnego, umieszczonego na liście parametrów wywoływanych przez wartość
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 lub EQUIV nie jest typu BOOLEAN
34	prawy argument operatora logicznego nie jest typu BOOLEAN
35	ograniczenie dolne w deklaracji tablicy nie jest typu arytmetycznego
36	ograniczenie górne w deklaracji tablicy nie jest typu arytmetycznego
37	wartość ograniczenia górnego jest mniejsza od wartości ograniczenia dolnego
38	wartość indeksu większa od 32767
39	identyfikator określa niewłaściwą klasę wyrażenia indeksowego
40	jak 39
41	błędnie podane wartości indeksów, element tablicy wykracza poza zarezerwowany dla niej obszar
42	liczba indeksów nie jest zgodna z deklaracją odpowiedniej tablicy

Nr błędu	Opis błędu
43[1)]	typ parametru aktualnego niezgodny ze specyfikacją parametru formalnego - REAL ARRAY albo INTEGER ARRAY - umieszczonego na liście parametrów wywoływanych przez wartość
44[1)]	typ parametru aktualnego niezgodny ze specyfikacją BOOLEAN ARRAY parametru formalnego, umieszczonego na liście parametrów wywoływanych przez wartość
45	niewłaściwy parametr aktualny funkcji standardowej; na przykład tekst w instrukcji wyjścia liczbowego nie jest formatem liczbowym
46	niewłaściwie określony parametr aktualny reprezentujący numer wejścia-wyjścia symbolicznego w instrukcjach procedur INP, OUT lub OUTCHAR
47	niewłaściwy parametr aktualny określający element wejścia-wyjścia
48	parametr aktualny procedury OUT nie jest tekstem
49	niewłaściwa liczba parametrów aktualnych funkcji standardowej
50	błąd w wywołaniu procedury wyjścia OUT
51	błąd w wywołaniu procedury wejścia INP
52	niepoprawne wyrażenie sterujące
57	dzielnik jest zerem

1) patrz odnośnik na stronie E-1

Nr błędu	Opis błędu
58	przekroczenie zakresu liczb
61	użycie procedury OUT, INP lub OUTCHAR jako parametru aktualnego
62	użycie instrukcji procedury OUT w treści procedury obliczającej wartość odwołania funkcyjnego występującego w wyrażeniu arytmetycznym lub boolowskim - będącym elementem wyjścia
78	argument, dla którego ma być obliczona wartość funkcji SQRT jest mniejszy od 0
79	za duży argument, dla którego ma być obliczona wartość funkcji EXP
80	argument, dla którego ma być obliczona wartość funkcji LN jest ujemny lub równy zero
87	błąd przy wprowadzaniu danych liczbowych lub logicznych
88	w danych brak elementu wejścia, który powinien być wartością logiczną
89	błąd w danych logicznych
90	błąd w danych liczbowych
91	liczba za duża dla maszyny
92	błąd w danych liczbowych
93	za dużo symboli w formacie rozmieszczenia
94	cudzysłów "'" nie rozpoczyna tekstu

Nr błędu	Opis błędu
95	błąd w formacie liczbowym; na przykład dwa razy występuje litera E oddzielająca mantysę od wykładnika lub za dużo wypisywanych cyfr
96	niepoprawny symbol rozpoczynający lub kończący format
97	niepoprawny symbol w formacie
98	niepoprawny symbol w formacie logicznym
99	za duży numer wyjścia symbolicznego

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
322	przekroczony obszar BEB
400	błąd przesłania z BEB