

6.11. Wejście z maszyny do pisania

Możliwość wprowadzania do maszyny fragmentów programu czy danych liczbowych przez ręczne pisanie na maszynie do pisania pojawia się w maszynie GIER na skutek przyciśnięcia na maszynie do pisania klawisza "t" w dwu sytuacjach:

I. W sytuacji "algol".

II. Po odczytaniu przez maszynę kodu END CODE, wydrukowaniu przez nią automatycznie sygnału

pause

i następnie zatrzymaniu się.

W obu przypadkach po przyciśnięciu klawisza "t" maszyna czeka na tekst z maszyny do pisania. Wtedy każdy wiersz napisany na maszynie i zakończony powrotem karetki (tzn. CAR RET) jest włączany do programu. Wiersz, którego pisanie rozpoczęto ale jeszcze nie zakończono uderzeniem klawisza CAR RET, może być skasowany w następujący sposób:

1) uderzamy na maszynie do pisania klawisze: LOWER CASE
UPPER CASE
LOWER CASE
UPPER CASE

albo

UPPER CASE
LOWER CASE
UPPER CASE
LOWER CASE

na co maszyna reaguje wydrukowaniem na czerwono symbolu

<

2) uderzamy klawisz "n", na co maszyna reaguje dopisaniem do poprzedniego znaku liter "no", co daje łącznie zapis

<no

Oznacza to, że wiersz ostatni (ale nie zakończony przez CAR RET) został skasowany i maszyna czeka na nowy tekst w jego miejsce.

W każdym momencie można powrócić do czytania na taśmie. W tym celu:

1) uderzamy - jak poprzednio - klawisze: LOWER CASE
UPPER CASE
LOWER CASE
UPPER CASE

albo

UPPER CASE
LOWER CASE
UPPER CASE
LOWER CASE

na co maszyna reaguje wydrukowaniem na czerwono symbolu

<

2) uderzamy klawisz "y", na co maszyna reaguje dopisaniem do poprzedniego znaku liter "yes", co daje łącznie zapis

<yes

i natychmiast przechodzi do odczytywania taśmy papierowej. Należy pamiętać, że maszyna tutaj nie zatrzymuje się. Dlatego

przed przyciśnięciem klawisza "y" należy sprawdzić, czy na czytniku taśma papierowa jest gotowa do czytania. Cały tekst wypisany ostatnio na maszynie do pisania zostaje przy tym włączony do programu.

6.12. Sygnały błędów składniowych programu

Translator GIER-ALGOLu sygnalizuje wykryte błędy składniowe programu w czasie trwania jego tłumaczenia. Sygnał alarmowy jest drukowany czerwono i składa się z czterech części:

numer
line
numer
nazwa błędu

Pierwszy numer jest tzw. numerem przejścia programu. W czasie tłumaczenia programu jest on przebiegany ośmiokrotnie, a więc pierwszy numer może mieć wartość od 1 do 8. Po pierwszym numerze jest drukowany wyraz "line", co znaczy wiersz. Po "line" zostaje wydrukowany numer wiersza programu tłumaczonego, w którym został wykryty błąd (maszyna liczy wiersze programu tłumaczonego poprzez liczenie kodów powrotu karetki, z tym że wiersz z pierwszym "begin" jest oznaczony numerem 0). Błędy wykryte w przebiegu 7 albo 8 są zawsze sygnalizowane dla linii 0. Sygnał kończy się nazwą błędu.

Poniżej omówimy możliwe nazwy błędów, klasyfikując je według pierwszego numeru sygnału alarmowego, tzn. wg numeru przejścia programu, a w zakresie tego samego przejścia, wg alfabetu.

1. character

na taśmie papierowej na wejściu pojawił się kod nie odpowiadający żadnemu symbolowi.

1. comment

wyraz comment nie jest poprzedzony ani przez begin, ani przez średnik.

1. compound

w danym wierszu jest ciąg symboli stanowiących początek określonego wyrazu algolowskiego lub symbolu złożonego, lecz brakuje niektórych dalszych symboli (np. begi zamiast begin).

1.) <improper>

ciąg symboli

) nazwa złożona z samych liter

nie jest zakończony :

1. string

po symbolu < nie następuje ani symbol < ani wzorzec.

1. sum

kod następujący na taśmie po SUM CODE nie jest zgodny z wartością obliczoną w czasie wprowadzania taśmy z programem.

2. too many identifiers

program używa za dużo (jak na pamięć maszyny) nazw (tzn. nazw zmiennych, etykiet, przełączników itd.) albo nazwy są za długie.

3. - delimiter

brak symbolu albo podkreślonego wyrazu algolowskiego oddzielającego dwie części wyrażenia (tzn. nazwy, liczby, wartości logiczne, łańcuchy, wyrażenia w nawiasach). Na przykład:

7.3 sin(5) między 7.3 i sin(5) brak operatora arytmetycznego albo nawiasu okrągłego, albo znaku relacji itp.

4 true między 4 i true brak operatora logicznego albo nawiasu, albo wyrazu if itp.

r.77 między r i liczbą .77 brak operatora arytmetycznego, nawiasu lub innego znaku,

r<<string> między r i łańcuchem <<string> brak przecinka lub ciągu znaków nazwa złożona z samych liter :(

Należy zwrócić uwagę na to, że wyraz "delimiter" jest w omawianym sygnale poprzedzony minusem.

3. delimiter

symbol albo podkreślony wyraz algolowski służący do rozdzielania nazw, liczb, wartości logicznych, łańcuchów, wyrażenia w nawiasach, jest użyty w niedopuszczalnej strukturze, albo też operator arytmetyczny nie jest poprzedzony prawidłowymi symbolami. Na przykład:

begin r(i:= znak (występuje po lewej stronie znaku :=

if go to po if w Algolu nie mamy nigdy bezpośrednio go to

if for po if w Algolu nie mamy nigdy bezpośrednio for

i:= xr; przed znakiem x brak zmiennej lub liczby.

3. number

błędny zapis liczby albo liczba za duża na maszynę. Podobny sygnał może pojawić się również w 7. przejściu programu.

3. - operand

(należy zwrócić uwagę na minus przed wyrazem "operand") jakaś część wyrażenia została opuszczona na końcu konstrukcji, jak np. r:= r/; po znaku / brak czegoś.

3. operand

nazwa albo liczba, albo wartość logiczna, albo łańcuch występuje w złym kontekście albo zostały opuszczone, jak np.

7:= liczba po lewej stronie znaku :=,
begin true wyraz true nie bywa nigdy pisany
 po begin,

a:= [i] brak nazwy tablicy przed indeksem.

3. termination

nieprawidłowe zakończenie struktury, jak np.
begin r:= a+b, zakończenie przecinkiem za-
 miast średnika,
 p(i, r; opuszczenie nawiasu zamykającego.

3. stack

przekroczenie możliwości translatora np. przez nadmierną ilość nawiasów typu begin ... end. Podobny sygnał można otrzymać w czasie przejścia nr 4,5 i 8.

4. stack

jak wyżej

5. - declar.

nazwa jest użyta w miejscu, gdzie nie była zadeklarowana. W przypadku tego sygnału możemy mieć błędnie podany numer wiersza, a mianowicie:

jeśli dana nazwa jest parametrem aktualnym, to zostaje podany numer wiersza, w którym był otwarty nawias zawierający daną listę parametrów aktualnych,

jeśli dana nazwa jest nazwą przełącznika, to podany numer wiersza odnosi się do miejsca, gdzie był wyraz "begin" bloku, w którym dany przełącznik został użyty.

5. + declar.

ta sama nazwa została dwukrotnie zadeklarowana w tym samym bloku, albo występuje dwukrotnie na tej samej liście parametrów formalnych; również ta sama etykieta mogła być użyta dwukrotnie w tym samym bloku, powodując dwuznaczność.

5. - formal

nazwa występuje w specyfikacjach, a nie występuje jako parametr formalny.

5. - specific.

nazwa występuje jako parametr formalny, a nie jest specyfikowana.

5. + specific.

ta sama nazwa jest dwukrotnie specyfikowana w tej samej deklaracji procedury.

5. value

w deklaracji procedury na liście parametrów value występuje parametr, nie mogący mieć wartości.

5. stack

lista nazw przewyższa możliwości translatora. Podobny sygnał możemy otrzymać w przejściu nr 3, 4 i 8.

6. subscript 704

liczba indeksów użyta dla zmiennej indeksowanej nie odpowiada liczbie indeksów w deklaracji danej tablicy.

6. proc. call or ident. 790

nazwa procedury występująca przed otwierającym nawiasem (jest zła albo odpowiada złej ilości parametrów).

6. proc. call or ident. 840

nazwa procedury występuje w kontekście niezgodnym z jej deklaracją.

6. type 576

po operatorze arytmetycznym następuje wyrażenie niewłaściwego typu, np. nazwa przełącznika albo zmienna booleowska.

6. type 582

po operatorze dzielenia algebrycznego następuje wyrażenie niewłaściwego typu, np. zmienna typu real czy boolean.

6. type 585

na zmienną booleowską jest podstawiana wartość wyrażenia niewłaściwego typu, np. wartość liczbowa.

6. type 590

po znaku relacji następuje wyrażenie niewłaściwego typu, np. po znaku = lub > zmienna booleowska.

6. type 593

po operatorze logicznym następuje wyrażenie niewłaściwego typu, np. po znaku \wedge liczba lub zmienna liczbowa.

6. type 596

po obu stronach operatora występują wyrażenia niewłaściwego typu, np. $i \vee a_1$, gdzie i jest zmienną typu integer, a a_1 nazwą tablicy.

6. type 599

na zmienną typu real jest podstawiana wartość wyrażenia niewłaściwego typu, np. przełącznik, czy wartość logiczna.

6. type 604

na zmienną typu integer jest podstawiana wartość wyrażenia niewłaściwego typu, np. procedura niefunkcyjna, czy wartość logiczna.

6. type 616

argument funkcji standardowej lub standardowej procedury pseudofunkcyjnej jest niewłaściwy, np. jako argumentu funkcji \ln użyto nazwy przełącznika lub tablicy.

6. type 630

po go to coś niewłaściwego albo w deklaracji przełącznika na liście jest parametr niewłaściwy, np. go to b, gdzie b jest zmienną booleowską, lub np. switch S:= r,... gdzie r jest zmienną liczbową.

6. type 633

między średnikami jest coś niewłaściwego, np. ; r ; gdzie r jest zmienną liczbową.

6. type 640

przed operatorem jest wyrażenie niewłaściwego typu, np. b= , gdzie b jest zmienną booleowską, która nigdy nie poprzedza operatora = , albo r\ , gdzie r jest zmienną liczbową.

6. type 646

przed otwierającym nawiasem kwadratowym zamiast nazwy tablicy jest coś innego.

6. type 649

w wywoływaniu procedury to drum albo from drum parametrem nie jest nazwa tablicy lecz coś innego.

6. type 657

po wyrazie then następuje wyrażenie niewłaściwego typu.

6. type 677

po obu stronach wyrazu else są niezgodzające się wyrażenia, np. 2-r else b, gdzie 2-r jest wyrażeniem liczbowym, a b zmienną booleowską. Ten sam sygnał otrzymujemy, jeśli po obu stronach konstrukcji else if ... then występują wyrażenia niezgodne.

6. type 686

między dwoma kolejnymi znakami := występuje zmienna niewłaściwego typu, np. r:= b:= 2, gdzie b jest zmienną booleowską.

6. type 690

po wyrazie for a przed do występuje zmienna lub wyrażenie niewłaściwego typu.

6. type 697

przed nawiasem kwadratowym zamykającym występuje wyrażenie niewłaściwego typu.

6. type 714

przed znakiem := występuje zmienna niewłaściwego typu.

6. type 725

przed wyrazem then albo między wyrazami while i do występuje wyrażenie niewłaściwego typu.

6. type 728

między for i := występuje zmienna niewłaściwego typu.

6. type 732

indeks jest niewłaściwego typu.

6. type 735

w deklaracji tablicy przed dwukropkiem w nawiasie kwadratowym jest wyrażenie niewłaściwego typu.

7. number

wyrażenie arytmetyczne zbudowane tylko z liczb i operatorów arytmetycznych ma wartość za dużą dla maszyny.

8. stack

w programie występuje równolegle za dużo (jak na możliwości translatora) etykiet, instrukcji "dla" i instrukcji warunkowych.

Oprócz powyższych sygnałów może jeszcze w każdym z przejść pojawić się sygnał

program too big

co oznacza, że program jest za duży na maszynę.

6.13. Poprawianie błędów składniowych. wykrytych przez maszynę w programie

Sygnały alarmujące błąd składniowy w programie - jak już o tym była mowa - podają m.in. numer wiersza, w którym dany błąd został wykryty. Ułatwia to bardzo odnalezienie błędu. Niemniej jednak zdarzają się przypadki, w których odszukanie błędu wskazanego przez maszynę nie jest proste. Bez wątpienia dużą rolę odgrywa tu wprawa. Należy jednak zwrócić uwagę, że mogą powstać sytuacje, w których maszyna alarmuje bardzo dużo błędów składniowych, mimo że został w gruncie rzeczy popełniony tylko jeden. Sytuacje takie powstają, gdy błąd popełniony wpływa w sposób istotny na sposób interpretowania dalszego ciągu programu przez maszynę. Jeśli na przykład w jakimś wyrażeniu arytmetycznym zostanie opuszczony nawias zamykający, to maszyna będzie interpretowała następujący po nim tekst jako dalszy ciąg wyrażenia w nawiasie, co może dawać najróżniejsze niespodzianki w składni i sygnalizację wielu nieprawidłowości. Podobna sytuacja może powstać również w przypadku opuszczenia średnika i interpretowania przez maszynę następującego po nim tekstu na przykład jako dalszego ciągu poprzedniej instrukcji. W takich przypadkach wstawienie jednego brakującego znaku powoduje nieraz zniknięcie wielu sygnałów alarmowych w czasie ponownej translacji.

Brak sygnałów alarmowych i pojawienie się sygnału "run" nie oznacza bezwzględnej poprawności programu. Maszyna nie sygnalizuje bowiem błędów merytorycznych programu, a jedynie błędy formalne. Nie są zatem sygnalizowane ani błędy w metodzie numerycznej, ani błędy w wartościach danych, ani błędy w organizacji programu itp.

6.14. Próbne liczenie

Z chwilą znalezienia się maszyny w sytuacji "run" można rozpocząć liczenie programu. Zanim to nastąpi, często przeprowadza się krótkie próbne liczenie, które może pozwolić wykryć ewentualne błędy merytoryczne w programie czy metodzie numerycznej. Próbne liczenie uruchamiamy tak jak liczenie normalne (patrz paragraf następny), z tym że należy przewidzieć w programie możliwość szybkiego otrzymania pierwszych rezultatów, aby dla ich otrzymania wystarczył krótki, kilkuminutowy czas, jaki zazwyczaj jest przeznaczony na uruchamianie programu.

Najczęściej błędy merytoryczne w programie czy w metodzie numerycznej powodują bardzo wyraźne zniekształcenia rezultatów. Spotyka się jednak czasem i przypadki, w których wyniki są zgodne z przewidywaniami, a jednak nie są poprawne na skutek popełnionych błędów merytorycznych. Nie ma tu ogólnych metod wykrywania możliwych błędów. Zależą one od rodzaju zagadnienia, jak również doświadczenia, inteligencji i wprawy rachującego.

6.15. Wykonanie obliczeń

Rozpocząć obliczenia można z chwilą pojawienia się sygnału

run

W tej sytuacji mamy przede wszystkim możliwość wyboru urządzenia wyjściowego, na którym chcemy otrzymać wyniki.

Jeśli w sytuacji "run" przyciśniemy na maszynie do pisania jakikolwiek klawisz z wyjątkiem

b, w, p, e,

(najczęściej przyciskamy tu klawisz SPACE jako najwygodniejszy), maszyna zacznie liczyć według programu, przyporządkowując instrukcjom wyjściowym typu "out-" perforator, a instrukcjom typu "write-" maszynę do pisania. Przyciśnięcie któregokolwiek z klawiszy "b", "w", "p" również uruchamia liczenie, ale z innym wykorzystaniem urządzeń wyjściowych: w przypadku przyciśnięcia "b" zarówno instrukcje "out-", jak i "write-" powodują wyprowadzanie wyników zarówno na perforator, jak i na maszynę do pisania; w przypadku przyciśnięcia "w" wszystkie instrukcje wyjściowe powodują druk na maszynie do pisania; w przypadku przyciśnięcia "p" wszystkie instrukcje wyjściowe powodują perforowanie wyników na taśmie papierowej na perforatorze. Wreszcie przyciśnięcie "e" powoduje przeskok do końca programu z wydrukowaniem końcowego sygnału "end".

Do sytuacji "run" można wrócić w dowolnym momencie liczenia programu. Będzie o tym mowa w paragrafie następnym.

Również w dowolnym momencie liczenia można przejść do sytuacji, w której przez przyciśnięcie któregokolwiek z klawiszy "b", "w", "p" można zmienić wybór urządzenia wyjściowe-

go na inny lub skoczyć do końca programu. Dokonuje się tego przez zmianę położenia klucza KA na małym pulpicie maszyny na przeciwne (tzn. przez włączenie klucza wyłączanego lub wyłączenie włączonego). Maszyna reaguje na to zatrzymaniem się (po kilku, kilkunastu, w niektórych wyjątkowych przypadkach po 20-30 sekundach). Wybieramy wtedy urządzenie wyjściowe lub skaczemy do końca programu przez przyciśnięcie odpowiedniego klawisza, jak wyżej. Przyciśnięcie klawisza różnego od "b", "w", "p", "e" powoduje - jak poprzednio - przyporządkowanie instrukcjom wyjściowym typu "out-" perforatora, a instrukcjom typu "write-" maszyny do pisania. Powyższy wybór jakiegokolwiek klawisza (z wyjątkiem oczywiście "e") uruchamia również od razu dalsze liczenie programu.

Liczenie programu przez maszynę (oczywiście przy założeniu jej poprawnego działania) kończy się zawsze jakimś sygnałem, po wydrukowaniu którego maszyna zatrzymuje się. Możliwe są tutaj następujące sygnały:

- p>end Maszyna zakończyła liczenie programu i doszła do jego końcowego
- end
- .
- alas Program wymaga więcej miejsc w pamięci operacyjnej niż na to pozwala jej pojemność (za dużo jest zmiennych, etykiet, instrukcji "dla" itp.).

array Program używa zadeklarować tablicę za dużą dla maszyny albo tablicę o ujemnej liczbie elementów.

exp Funkcja standardowa exp została wywołana z za dużym argumentem albo wynik potęgowania \uparrow z wykładnikiem typu real jest za duży dla maszyny (por. paragraf 1.15).

gier Standardowa procedura gierproc albo gierdrum czytania niewłaściwą taśmę albo kontrola sumowa na czytanej taśmie się nie zgadza.

index Wartość wyrażenia podanego jako indeks wykracza poza granice ustalone w deklaracji tablicy. Testowana jest tutaj tylko ostateczna wartość indeksu, tzn. jeśli w tym wyrażeniu występują jakieś inne indeksy dla zmiennych występujących w tym wyrażeniu, to nie są one testowane.

ln Funkcja standardowa ln została wywołana dla ujemnego argumentu. (Dla $\ln(0)$ nie otrzymujemy tego sygnału, ale wartość $-9.35_{10}49$). Ten sam sygnał otrzymujemy w przypadku pojawienia się potęgi o wykładniku real z liczby ujemnej.
 Uwaga: wartość ujemna mogła powstać zamiast oczekiwanej nieujemnej na skutek zaokrągleń w kolejnych wynikach działań arytmetycznych (por. paragraf 1.15).

param Procedura standardowa została wywołana z niewłaściwą ilością parametrów.

spill Liczona wartość liczbowa przekroczyła możliwość maszyny. Należy zapamiętać, że potęga o wykładniku ujemnym jest zawsze liczona najpierw dla wykładnika dodatniego, a potem jest brana odwrotność wyniku. Może się zatem zdarzyć, że dla ujemnego wykładnika potęgi otrzymamy sygnał spill, chociaż (w granicach dokładności maszyny) powinna być otrzymana wartość zero.

sqrt Funkcja standardowa sqrt (pierwiastek kwadratowy) została wywołana dla ujemnego argumentu.

Uwaga: wartość ujemna mogła powstać zamiast oczekiwanej nieujemnej na skutek zaokrągleń w kolejnych działaniach arytmetycznych.

drum alas Procedura standardowa to drum albo from drum została wywołana dla wartości drumplace przekraczającej możliwości bębna.

Jeśli po otrzymaniu któregośkolwiek z wyżej wymienionych sygnałów i zatrzymaniu się maszyny uderzymy na maszynie do pisania klawisz "r" maszyna wraca do sytuacji "run", drukując sygnał "run" i zatrzymując się.

Uderzenie zamiast "r" jednego z klawiszy "b", "w", "p" powoduje wyprowadzenie na wskazane tym klawiszem urządzenie wyjściowe aktualnych wartości zmiennych zawartych w pamięci operacyjnej. Będziemy to nazywali krótko wyprowadzeniem z a w a r t o ś c i p a m i ę c i o p e r a c y j n e j. Będzie o tym mowa w paragrafie 6.19.

Jeżeli zamiast wymienionych wyżej klawiszy uderzymy jakimkolwiek inny, maszyna wraca do sytuacji "algol", drukując sygnał "algol" (ewentualnie "algol KA", "algol KB" albo "algol KC") i zatrzymuje się. Wyjątek stanowi przypadek, gdy program przez zajęcie zbyt dużo miejsc na bębnie zniszczył część translatora, czyniąc go niezdadnym do dalszego użycia. W takim przypadku zamiast sygnału "algol" otrzymujemy sygnał

gone

oznaczający konieczność ponownego wprowadzenia translatora do maszyny, o co należy się zwrócić do dyżurującego przy maszynie technika.

W przypadkach dużych obliczeń spotykamy się z koniecznością wielokrotnego liczenia tym samym programem i tym samym koniecznością wielokrotnego wprowadzania programu do maszyny. Aby uniknąć przy tym wielokrotnego powtarzania translacji tego samego programu z języka GIER-ALGOL na kod wewnętrzny maszyny, można wyprowadzić z maszyny zawartość bębna magnetycznego z przetłumaczonym już programem i później wprowadzać bardzo szybko gotowy już program bez jego tłumaczenia. Będzie o tym mowa w paragrafie 6.21.

Dobrze jest orientować się przynajmniej w przybliżeniu w czasach potrzebnych do wykonania przez maszynę różnych operacji, aby - w razie potrzeby - programować, oszczędzając czas pracy maszyny. Będzie o tym mowa w paragrafie 6.17.

W paragrafie następnym omówimy jeszcze sposoby sprowadzania maszyny do sytuacji "run".

6.16. Sprowadzanie maszyny do sytuacji "run"

Jak o tym już była mowa, sytuacją "run" nazywamy sytuację, gdy maszyna wydrukowała sygnał

run

i zatrzymała się. Oznacza to, że w maszynie jest program przetłumaczony na kod wewnętrzny i maszyna jest gotowa go liczyć (patrz paragraf poprzedni).

Sytuacja "run" powstaje automatycznie po zakończeniu wprowadzania programu i jego translacji, gdy maszyna nie znalazła błędu formalnego w tym programie (w przeciwnym razie maszyna wraca do sytuacji "algol").

Często zdarza się, że chcemy wrócić do sytuacji "run" w czasie liczenia programu. Tak jest na przykład w przypadku, gdy nie jesteśmy zadowoleni z dotychczasowych wyników liczenia i chcemy rachunek powtórzyć, zmieniając wartości niektórych parametrów. Do sytuacji "run" można wrócić w dowolnym momencie liczenia programu.

Rozróżnimy dwa przypadki:

I. Maszyna wydrukowała jeden z następujących sygnałów

| | | |
|-----------|-------|-------|
| alas | exp | param |
| array | gier | spill |
| drum alas | index | sqrt |
| end | ln | |

i zatrzymała się (znaczenie wymienionych sygnałów zostało omówione w paragrafie poprzednim). W takim przypadku przejście do sytuacji "run" uzyskujemy uderzając na maszynie do pisania klawisz "r".

II. W pozostałych przypadkach można przejść do sytuacji "run" na dwa sposoby. Pierwszy polega na spowodowaniu przekroczenia możliwości maszyny i wydrukowania sygnału "spill", po czym wystarczy - jak w przypadku I - uderzyć klawisz "r". Drugi sposób polega na zmianie położenia klucza KA na małym pulpicie maszyny (tzn. wyłączeniu klucza włączonego albo włączeniu wyłączonego). Omówimy je kolejno.

Spowodować sygnał "spill" można następująco:

- 1) na dużym pulpicie maszyny przycisnąć klawisz MIKRO-TEMPI-STOP,
- 2) przycisnąć na tymże pulpicie klawisz r1 (tzn. włączyć rejestr r1),
- 3) przycisnąć czarny klawisz zupełnie z lewej strony u dołu dużego pulpitu (tzn. wyłączyć rejestr r1),
- 4) przycisnąć klawisz NORMAL-START, na co maszyna reaguje wydrukowaniem sygnału

spill

W drugim sposobie zmieniamy położenie klucza KA na małym pulpicie przez przyciśnięcie odpowiedniego z dwu klawiszy przy napisie KA. Jeśli paliło się przy nim zielone światło, to powinno zgasnąć, jeśli nie paliło się, powinno się zapalić. Po upływie kilku sekund (w wyjątkowych przypadkach po 20-30 sekundach) maszyna zatrzymuje się. Wtedy na maszynie do pisania uderzamy klawisz e, na co maszyna reaguje wydrukowaniem sygnału

end

Wtedy - jak w przypadku I - wystarczy na maszynie do pisania uderzyć klawisz "r", by otrzymać sytuację "run", którą maszyna sygnalizuje drukiem sygnału

6.17. Czas pracy maszyny

Czas wykonania programu przez maszynę nie zależy tylko od składników algorytmu, ale również od stopnia wypełnienia pamięci operacyjnej, od ilości indeksów w tablicach, od ilości instrukcji "dla". Niemniej jednak może być dla programującego pożyteczne znać przeciętne czasy wykonywania niektórych typowych operacji na maszynie GIER. Oto one:

| Operacja wykonywana przez maszynę | Czas wykonania w milisekundach |
|--|-----------------------------------|
| Dodawanie lub odejmowanie | 0.12 |
| Mnożenie | 0.18 |
| Dzielenie | 0.21 |
| Podniesienie do kwadratu | 0.18 |
| Podniesienie do trzeciej potęgi | 0.4 |
| Podniesienie do dziesiątej potęgi | 5.5 |
| Podniesienie do setnej potęgi | 8 |
| Podniesienie do potęgi o wykładniku <u>real</u> | 12 |
| Najprostszy warunek "jeśli" | 0.3 |
| Wywołanie zmiennej z jednym prostym indeksem | 0.9 |
| Wywołanie zmiennej z dwoma prostymi indeksami | 1.2 |
| Obliczenie nowej wartości zmiennej kontrolowanej w instrukcji "dla" w przypadku elementu <u>step 1 until n</u> | 0.6 |
| Otwarcie i zamknięcie bloku z deklaracją zmiennych nieindeksowanych | 1.4 |
| Otwarcie i zamknięcie bloku z deklaracją tablicy | 3.0 |
| Wykonanie instrukcji podstawienia $a:=0$ | 0.05 |
| Wykonanie instrukcji podstawienia $a:=b$ | 0.1 |
| Wykonanie skoku do etykiety <u>go to A</u> | 0.8 |
| Wykonanie skoku poprzez przełącznik <u>go to S[i]</u> | 2.1 |
| Wywołanie procedury standardowej: | |
| abs | 0.17 |
| arctan | 6.6 |
| cos | 6.0 |
| entier | 0.4 |
| exp | 5.8 |
| ln | 5.6 |
| sign | 3.2 |
| sin | 5.8 |
| sqrt | 6.2 |
| kbon (b:=kbon) | 3.5 |
| Wywołanie procedury niestandardowej, deklarowanej: | |
| bezparametrowej | 3.8 |
| z 1 parametrem | 4.7 |
| z 2 parametrami | 5.2 |
| z 3 parametrami | 5.5 |

Jak widać z powyższego wykazu, operowanie tablicami jest bardzo czasochłonne.

6.18. Zadanie

Za pomocą programu z przykładu 5.39 obliczyć na maszynie GIER pierwiastki podanego tam układu równań, wychodząc z wartości $x=0$, $y=-2$, $h=0.1$, a następnie stosując $h=0.01$, 0.001 , 10^{-4} , 10^{-5} , 10^{-6} , 10^{-7} i 10^{-8} .

Sprawdzić, że takie same wyniki otrzymujemy wychodząc z punktów:

$x=4$, $y=-4$,
 $x=-4$, $y=-4$.

Powtórzyć obliczenia dla punktów wyjściowych:

$x=4$, $y=4$,
 $x=-4$, $y=4$,
 $x=0$, $y=0$,

i wytłumaczyć otrzymane rezultaty.

6.19. Wyprowadzanie zawartości pamięci operacyjnej

Sytuacja, w której można wyprowadzić zawartość pamięci operacyjnej, powstaje po wydrukowaniu przez maszynę któregoś z sygnałów:

| | | |
|-----------|-------|-------|
| alas | exp | param |
| array | gier | spill |
| drum alas | index | sqrt |
| end | ln | |

Wyprowadzenie zawartości pamięci operacyjnej ma za zadanie ułatwienie rozpoznania przyczyny pojawienia się takiego sygnału. Przyciśnięcie wtedy któregoś z klawiszy "b", "p", "w" na maszynie do pisania powoduje uruchomienie programu wyprowadzającego zawartość pamięci operacyjnej, z tym że przyciśnięcie "b" powoduje wyprowadzenie na perforator i na maszynę do pisania, przyciśnięcie "p" wyprowadzenie tylko na perforator, przyciśnięcie "w" wyprowadzanie tylko na maszynę do pisania. Już w czasie wyprowadzania zawartości pamięci operacyjnej można również zmieniać urządzenie wyjściowe. W tym celu należy zmienić położenie klucza KA na małym pulpicie (tzn. włączyć wyłączony albo wyłączyć włączony) i po zatrzymaniu się maszyny (po kilku sekundach) przycisnąć odpowiedni klawisz "b", "p" albo "w". Przyciśnięcie tutaj klawisza "e" powoduje zakończenie wyprowadzania zawartości pamięci operacyjnej i druk sygnału "end". Przyciśnięcie jakiegokolwiek innego klawisza uruchomi dalsze wyprowadzanie na tym samym, poprzednio wskazanym urządzeniu wyjściowym.

Wyprowadzanie następuje na skutek wykonania specjalnego programu włączonego do translatora i zostaje wykonane w formie łatwej do odczytania, ponieważ są wyprowadzane również nagłówki, stanowiące rodzaj komentarza dla następujących po nich liczb. Wyprowadzeniu z pamięci operacyjnej podlegają:

1) W a r t o ś c i z m i e n n y c h:

Wartości typu integer są wyprowadzane według wzorca

↳-ndddddd↳

Wartości typu real są wyprowadzane według wzorca

↳-.ddd_n-ddd↳

Wartości zmiennych już zadeklarowanych, ale takich na które jeszcze nie dokonano żadnego podstawienia, są wyprowadzane w postaci wartości logicznych.

Wartości logiczne napisane explicite w programie w języku ALGOL są wyprowadzane jako 1 dla true i -1 dla false. Wartości logiczne utworzone przez wyrażenia są wyprowadzane albo jako + dla true a - dla false, albo jako liczby o wartości bezwzględnej większej od 1 dla true i mniejszej od 1 dla false.

2) Charakterystyczne punkty programu:

- wejścia do procedur,
- pojawienie się etykiety,
- punkty powrotu; na przykład punkty, do których maszyna powraca po wykonaniu wywołania procedury, albo punkty, do których maszyna powraca po obliczeniu wyrażenia stanowiącego parametr aktualny procedury.

Punkty charakterystyczne są wyprowadzane przez podanie odpowiadającego im adresu na bębnie. Adres ten jest podawany w formie dwu liczb całkowitych. Pierwsza z nich może mieć wartość od 0 do 319 i jest numerem tzw. ścieżki na bębnie, druga może mieć wartość od 0 do 39 i jest numerem miejsca na ścieżce (bęben ma 320 ścieżek po 40 miejsc, co daje właśnie 12 800 miejsc na bębnie). Z numeru ścieżki *s* i numeru miejsca *n* na ścieżce można obliczyć wartość drumplace dla danego miejsca bębna ze wzoru

$$\text{drumplace} := 40 \times s + n$$

Kolejność wyprowadzania jest następująca: Najpierw zostaje wyprowadzony sygnał

stack

po czym następują wartości zmiennych deklarowanych w programie z own (o ile takie, oczywiście były). Wartości zmiennych own są drukowane począwszy od tych, które były zadeklarowane w bloku najbardziej wewnętrznym, kolejno aż do tych, które były zadeklarowane w bloku najbardziej zewnętrznym. Dla każdego bloku wartości tych zmiennych są wyprowadzane w takim porządku, w jakim były zadeklarowane.

Pozostałe liczby są wyprowadzane z nagłówkami, które stanowią do nich komentarz. Są możliwe następujące nagłówki

block
points
variables
array
proc call
return
name call
value call

Nagłówek "block" oznacza wejście do bloku albo do ciała procedury (nawet, gdy ciało procedury nie jest blokiem). Należy tu konstrukcję programu rozpatrywać tak, jakby w programie w miejscach wywołania procedur były podstawiane ich ciała, tworzące w ten sposób nowe bloki wewnętrzne. Wyprowadzenie zawartości pamięci operacyjnej po zmiennych z own następuje począwszy od bloku najbardziej zewnętrznego. Po każdorazowym nagłówku "block" wyprowadzenie następuje w czterech częściach. Jeżeli dany blok jest ciałem procedury funkcyjnej, w części pierwszej zostaje wydrukowana wartość liczbowa tej procedury. Jeżeli wartość ta nie została jeszcze podstawiona, zostaje wyprowadzona jako 0. Jeżeli dany blok nie jest ciałem procedury funkcyjnej, pierwszej części nie ma w ogóle i po nagłówku "block" następuje od razu nagłówek "points" (czytaj: points; znaczy: punkty), otwierający część drugą. W części drugiej są wyprowadzane wszystkie punkty charakterystyczne lokalne dla danego bloku (wejścia do ciał procedur, punkty pojawienia się etykiet) w takiej kolejności, w jakiej są w programie. Część trzecia rozpoczyna się nagłówkiem "variables" (czytaj: weriebłs; znaczy: zmienne) albo "array" i zawiera wartości zmiennych lokalnych dla danego bloku, z tym że normalnie pierwszych kilka wartości odpowiada zmiennym roboczym wprowadzanym przez translator i nie interesującym programisty. Każdy ciąg zmiennych nieindeksowanych jest tu poprzedzony nagłówkiem "variables", a każdy ciąg wyrazów tablicy nagłówkiem "array". Kolejność wyprowadzania zmiennych zgodna z kolejnością ich deklaracji w programie, z tym że po nagłówku "array" następuje najpierw jeden wiersz, w którym są podane wymiary tablicy, tzn. liczby różnych wartości przyjmowanych przez kolejne indeksy tablicy. Na przykład 3x8 oznacza, że pierwszy indeks tablicy przyjmuje trzy różne wartości, a drugi osiem, tzn. tablica ma ogółem 24 wyrazy. Dopiero począwszy od następnego wiersza są wyprowadzane wartości wyrazów tablicy, w kolejności takiej, jaka była podana dla procedury input (por. paragraf 5.33), po 5 wartości w jednym wierszu. Na przykład wyrazy tablicy A zadeklarowanej jako

array A[0:2,1:8]

będą wyprowadzane w następującym szyku

3x8

| | | | | |
|--------|--------|--------|--------|--------|
| A[0,1] | A[0,2] | A[0,3] | A[0,4] | A[0,5] |
| A[0,6] | A[0,7] | A[0,8] | A[1,1] | A[1,2] |
| A[1,3] | A[1,4] | A[1,5] | A[1,6] | A[1,7] |
| A[1,8] | A[2,1] | A[2,2] | A[2,3] | A[2,4] |
| A[2,5] | A[2,6] | A[2,7] | A[2,8] | |

Uwaga: istnieje możliwość otrzymania błędnych wyników, a mianowicie w przypadku, gdy w programie bezpośrednio po deklaracji tablicy następuje zmienna typu integer o wartości będącej podzielnikiem liczby różnych wartości przyjmowanych przez pierwszy indeks tablicy.

Część czwartą stanowią wywołania procedur wewnątrz danego bloku. Każde wywołanie otrzymuje nagłówek "proc.call" (czytaj: prosz idź kol; znaczy: wywołanie procedury), po czym zostają wyprowadzone wartości wszystkich parametrów aktualnych wywoływanej procedury. Wartości te są poprawne dla parametrów wołanych przez wartość, a bezsensowne dla parametrów wołanych przez nazwy. Po tym następuje sygnał "return" (czytaj: riterń; znaczy: powrót) będący nagłówkiem wiersza, w którym zostaje wyprowadzony punkt programu następujący bezpośrednio po wywołaniu procedury. Z kolei zostaje wyprowadzony nagłówek "value call" (czytaj: welju kol; znaczy: wywołanie wartości) z podaniem po nim punktu, leżącego w pobliżu punktu wejścia do procedury (dokładne określenie tego punktu jest trudne). Jeśli procedura nie ma parametrów wołanych przez wartość, ani nagłówek "value call", ani następujący po nim punkt nie są oczywiście wyprowadzane. Po powyższym zostaje wyprowadzony nagłówek "block" i dane dotyczące ciała procedury. W momentach wywoływania parametrów przez nazwy zostaje wyprowadzony nagłówek "name call" (czytaj: nejm kol; znaczy: wywołanie przez nazwę) i po nim punkt powrotu bezpośrednio następujący po formalnym parametrze, na którym dokonuje się podstawienie przez nazwę.

Procedury standardowe sprawiają przy wyprowadzaniu specjalne trudności:

1) parametr procedur abs, arctan, cos, entier, exp, gier, ln, outchar, outcopy, outsp, setchar, sign, sin, sqrt, writchar, writecopy jest obliczany przed wywołaniem samej procedury,

2) procedury input, output, write, outtext, writetext z chwilą rozpoczęcia ich wykonania usuwają z pamięci operacyjnej swoje parametry,

3) procedury standardowe używane jako parametry aktualne innych procedur psują wyprowadzenie wywołania tych procedur (tzn. proc. call),

4) procedury to drum, from drum, to buf, from buf, to car, from car, split, pack, gierproc, gierdrum są wyprowadzane normalnie.

Wyprowadzanie kończy się w tym miejscu programu, w którym nastąpił jeden z sygnałów wymienionych na początku tego paragrafu. Jeśli był nim "spill" spowodowany przez potęgowanie z całkowitym wykładnikiem, ostatnim będzie nagłówek "proc. call" z następującą po nim wartością wykładnika potęgi.

Należy pamiętać, że w pamięci operacyjnej są przechowywane dane dotyczące bloków (czy ciał procedur), których wykonanie się rozpoczęło, ale jeszcze nie zakończyło. Po zakończeniu wykonania bloku jego dane są z pamięci operacyjnej wymazywane.

6.20. Zadanie

Następujący program:

begin real a,d,f; integer i,j; Boolean b1,b2;

array A[0:3,1:7], C[0:6];

procedure MN(T1,p,q,T2); value p; integer p,q; array T1,T2;


```

for i:= 0 step 1 until q do
    begin T2[i]:= 0;
        for j:= 0 step 1 until p do
            T2[i]:= T2[i]+T1[j,i+1] end;
for i:= 0,1,2,3 do for j:= 1,2,3,4,5,6,7 do A[i,j]:= i×j;
L1: MN(A,3,6,C);  a:= C[1]+C[3]+C[5];
d:= C[2]/C[4];
b1:= if C[2]>C[5] then true else false;
L2: b2:= if a<100 then false else b1;
Q: begin own real b;  procedure PR(x,y);  real x,y;  y:= x÷5;
integer k;
L3: for k:= 0,1,2,3,4,5,6 do PR(C[k],C[k]);
b:= C[2]+C[4]+C[6];
a:= if b1 then b else 0;
if b2 then go to L2 end;
f:= 1/a÷d
end;

```

uruchomić na maszynie GIER, a po otrzymaniu sygnału "spill" wyprowadzić zawartość pamięci operacyjnej na perforator. Po przedrukowaniu informacji wyperforowanych na taśmie papierowej na flexowriterze zinterpretować otrzymane dane.

6.21. Wyprowadzanie zawartości bębna

Zawartość bębna wyprowadza się poprzez perforator na taśmę papierową w formie binarnej (w systemie binarnym czyli dwójkowym), aby można było później wczytywać taśmę z powrotem do maszyny bez ponownego tłumaczenia z języka ALGOL na kod wewnętrzny, a zatem szybciej. Wyprowadzenie następuje dowolnymi fragmentami na skutek wykonania specjalnego programu pod nazwą "binout", włączonego do translatora i zawarte go na odrębnej taśmie papierowej, tzw. taśmie E. Normalnie programu "binout" na bębnie nie ma i musi być albo wprowadzony na bęben (wariant 3) albo czytany z taśmy (warianty 0,1 2,5). Program ten zostaje uruchomiony przez instrukcję w programie

$\text{gierproc}(\langle \text{binout} \rangle, W, c_1, d_1, c_2, d_2, \dots, c_n, d_n)$

gdzie W jest wyrażeniem arytmetycznym, którego wartością musi być 0, 1, 2, 3 albo 5, a c_i, d_i ($i=1, \dots, n$) są parami wyrażen arytmetycznych, których wartości dają odpowiednio wartość zmiennej drumplace , od której począwszy ma rozpocząć się wyprowadzenie i -go fragmentu z bębna, oraz liczbę miejsc na bębnie, jaka ma być w tym fragmencie wyprowadzona. W wariantach 0, 1, 2, 5 w momencie wywoływania powyższej instrukcji taśma E translatora musi być założona na czytnik.

Wartość wyrażenia W decyduje o wyborze jednego z 5 możliwych wariantów programu "binout". Mianowicie, jeżeli wartością wyrażenia W jest 0, zostają wyprowadzone jedynie fragmenty zawartości bębna wskazane kolejno parami wartości $(c_1, d_1), (c_2, d_2)$ itd. Jeżeli wartością W jest 1, oprócz wskazanych fragmentów zostaje również wyprowadzony przetłumaczony program, znajdujący się na bębnie. Jeżeli wartością W jest 2, to oprócz wskazanych fragmentów zostaje wyprowadzony nie tylko przetłumaczony program, ale także część translatora zawierająca sterowanie działaniem programu, program wyprowadzający zawartość pamięci operacyjnej i procedury standardowe. Jeżeli wartością W jest 3, zostają wyprowadzone tylko wskazane fragmenty, ale tak, że późniejsze ich wprowadzenie do maszyny jest możliwe tylko poprzez procedury gierproc lub gierdrum (gdy operujemy fragmentami programu napisanymi w kodzie wewnętrznym maszyny). Wreszcie jeżeli wartością W jest 5, oprócz wskazanych fragmentów zostaje wyprowadzony cały translator z bębna.

Na przykład, jeżeli chcemy wyprowadzić zawartość bębna począwszy od $\text{drumplace}=10\ 000$, miejsc 600 i począwszy od 8 400, miejsc 150, a ponadto przetłumaczony program, możemy zrobić to w ten sposób, że pisząc uprzednio program w języku ALGOL, umieszczamy w nim instrukcję

$\text{gierproc}(\langle \text{binout} \rangle, 1, 10000, 600, 8400, 150)$

Warianty 0, 1, 2, i 5 powodują automatyczne wyprowadzenie na samym początku specjalnego programu, który przy późniejszym wczytywaniu taśmy z powrotem do maszyny i przesyłaniu jej zawartości na bęben steruje tym wprowadzeniem. W wariantach 3 wprowadzaniem sterują procedury gierproc i gierdrum .

Wprowadzanie z powrotem na bęben zawartości taśmy papierowej wytworzonej przez "binout" w wariantach 0, 1, 2, 5 odbywa się (po założeniu tej taśmy na czytnik) następująco:

- 1) Włączamy na małym pulpicie klucze KA i KB,
- 2) Przyciskamy na małym pulpicie klawisz HP,
- 3) Jeżeli maszyna wydrukuję sygnał hp-knap, przyciskamy na maszynie do pisania kolejno klawisze 1 (tzn. piszemy 1); jeżeli maszyna wydrukuję FEJL albo KC ALGOL, przyciskamy klawisz odstepu.

Po wprowadzeniu zawartości taśmy z powrotem na bęben, maszyna przechodzi do sytuacji "run" w przypadku wariantów 0,

1,2, do sytuacji "algol" w przypadku wariantu 5, a liczy program dalej w przypadku wariantu 3.

Wyprowadzenie zawartości bębna wymaga około 170 swobodnych miejsc w pamięci operacyjnej, wprowadzenie z powrotem na bęben około 400 miejsc. Automatycznie przeprowadza się kontrolę sumową. Jeśli przy wczytywaniu z powrotem taśmy nastąpi niezgodność sumy kontrolnej, otrzymujemy zwykły sygnał

sum

i wtedy próbujemy wprowadzać taśmę jeszcze raz, przekładając ją na czytniku wstecz aż do początku i przyciskając na maszynie do pisania klawisz odstępu.

Jeśli użyta liczba parametrów w procedurze gierproc była nieparzysta, albo liczba miejsc w którymkolwiek wyprowadzanym fragmencie bębna wypadła ujemna, albo wreszcie wskazany fragment bębna przekroczył pojemność bębna, zostaje wydrukowany sygnał alarmowy

param

6.22. Zadanie

Następujący program:

```
begin integer c,i,j; integer array A,B[1:40];
if kb on then
  begin for i:= 1 step 1 until 40 do A[i]:= i;
  c:= drumplace; to drum(A);
  gierproc(<<binout>,1,c,40) end else
  begin drumplace:= c; from drum(B);
  for i:= 1,2,3,4,5 do begin writcr;
    for j:= 1 step 1 until 8 do
      write(<dddddd>,B[8*(i-1)+j]) end end end;
```

uruchomić na maszynie GIER, z tym że w sytuacji "run" najpierw włączyć klucz KB, a potem dopiero puścić w ruch program. Po otrzymaniu na perforatorze taśmy z zawartością bębna, założyć ją na czytnik i wprowadzić do maszyny tak, jak to było opisane w poprzednim paragrafie. Po otrzymaniu "run" włączyć klucze KA i KB i uruchomić program. Co wtedy zostanie wydrukowane przez maszynę?

6.23. Opracowywanie wyników obliczeń

Trudno podać ogólne reguły dla opracowywania wyników otrzymanych z maszyny, ale należy podkreślić konieczność przeprowadzenia dwu rzeczy:

- 1) kontroli otrzymanych wartości liczbowych,
- 2) starannego opisu otrzymanych wyników.

Istnieje wiele możliwych przyczyn powstawania błędów wyników. Może to być:

- błędne sformułowanie problemu,
- błędna metoda rozwiązywania problemu,
- błędny algorytm,
- błędy merytoryczne w programie,
- błędy przy dziurkowaniu programu na taśmie papierowej,
- błędy czytania taśmy przez czytnik maszyny,
- błędne działanie maszyny,
- błędy perforacji wyników przez maszynę,
- błędy przedruku taśmy perforowanej na flexowriterze.

Nie istnieją - niestety - ogólne metody wykrywania błędów. Na szczęście prawdopodobieństwo błędu w przypadkach znanych problemów, znanej maszyny, znanych pracowników można zredukować do pomijalnych wielkości, wypracowując w ciągu dłuższego okresu pracy na maszynie nieraz bardzo proste, ale wystarczające metody kontroli.

Przy pewnej wprawie programy pisze się tak, że wyprowadzone wyniki są już w pełni automatycznie opisywane. Gdy program takich opisów nie przewiduje, musimy je wprowadzać później ręcznie, na przykład atramentem. Należy jednak pamiętać, że opisy takie są konieczne, ponieważ nawet biegły rachmistrz może łatwo zgubić się w długich kolumnach liczb nie opatrzonych żadnymi komentarzami.

Dla poprawnej pracy na maszynie nie wystarcza z pewnością znajomość zasad programowania. Potrzebne są również wprawa i doświadczenie.