

i kolumny może być różny. Kolejność zapisu i czytania wyrazów tablicy jest jednak zawsze taka, jaka była opisana dla procedury input w paragrafie 5.33, tylko w odwrotnym porządku: początek tablicy dla najmniejszej wartości drumplace, koniec dla największej.

### 5.66. Zadanie

Tablicę A należy z pamięci operacyjnej przepisać na bęben, a następnie z bębna podstawić jej wyrazy do tablicy B w pamięci operacyjnej. Napisać odpowiedni fragment programu.

### 5.67. Zadanie

Tablice A, C, D i P zostały wprowadzone do programu deklaracją

```
array A[1:m,1:n], C[1:n], D[1:m], P[1:1]
```

Tablica A została przesłana z pamięci operacyjnej na bęben programem

```
c := drumplace;
```

```
to drum(A);
```

po czym została w pamięci operacyjnej wymazana.  
Jak należy zaprogramować:

- 1) podstawienie na P[1] wartości A[1,j],
- 2) podstawienie na C i-go wiersza tablicy A,
- 3) podstawienie na D j-ej kolumny tablicy A.

### 5.68. Zadanie

Rozwiązać zadanie poprzednie w przypadku, gdy tablica A jest za duża dla pamięci operacyjnej i jej deklaracja jak wyżej spowodowałaby wydrukowanie przez maszynę sygnału alarmowego

```
array
```

Wobec tego tablica A nie jest w ogóle deklarowana i wprowadzona na bęben z taśmy papierowej za pośrednictwem tablicy C programem

```

c:= drumplace;

for p:= 1 step 1 until m do
    begin input(C); to drum(C) end;

```

### 5.69. Zadanie

Metodą użytą w zadaniu 4.8 ułożyć program na rozwiązywanie dużych układów algebraicznych równań liniowych, aż do 80 równań z tyluż niewiadomymi, posługując się bębmem magnetycznym ( $80 \times 81 = 6480 < 7000$ , co pozwala zmieścić macierz współczynników na bębnie). Zakładamy, że macierz współczynników (włącznie z wyrazami wolnymi jako ostatnią kolumną) mamy wyperforowaną na taśmie papierowej, z tym że poprzedza ją liczba równań układu, wyperforowana jako pierwsza liczba na taśmie. Wyniki mają być wyperforowane na taśmie według wzorca  $\{+n.ddddd_0+d\}$  tak, aby późniejszy przedruk na flexowriterze dał je w pięciu pionowych kolumnach, zaznaczonych konikami tabulatora. Wyniki te mają mieć nagłówek:

Rozwiązanie układu równań:

### 5.70. Współpraca z buforem w przypadku tłumacza GIER ALGOL III BUF. Procedury to buf i from buf

Sposób korzystania z bufora za pośrednictwem procedur to buf i from buf jest analogiczny do opisanego wyżej sposobu korzystania z bębna za pośrednictwem procedur to drum i from drum. Używa się przy tym tej samej zmiennej standardowej drumplace. Różnice są jedynie następujące:

- na bufor i z bufora można przysyłać nie tylko tablice - jak to było w przypadku bębna magnetycznego - ale również zmienne nieindeksowane, tzn. że argumentem procedur to buf i from buf może być zarówno nazwa tablicy jak i nazwa zmiennej nieindeksowanej,

- bufor ma tylko 4096 miejsc numerowanych od 0 do 4095, z których każde przechowuje jedną liczbę; przekroczenie pojemności bufora nie powoduje sygnału alarmowego a, przeciwnie, przesyłanie trwa nadal tak, jakby idąc w kierunku malejących numerów, po numerach 1, 0 następowały znowu numery 4095, 4094 itd.; dzieje się to dlatego, że w przypadku otrzymania wartości drumplace spoza przedziału  $\langle 0, 4095 \rangle$  do procedur to buf i from buf wchodzi zamiast drumplace dodatnia reszta z podzielenia jej przez 4096, a więc na przykład zamiast drumplace = 4098 wchodzi drumplace = 2, zamiast drumplace = -31 wchodzi drumplace = 4065, zamiast drumplace = 8300 wchodzi drumplace = 108 itd.,

- procedury to buf i from buf są procedurami pseudofunkcyjnymi typu real a nie integer jak to drum i from drum; wywołanie ich jednak może być również w formie osobnej instrukcji, tak jak to było w przypadku to drum i from drum.

Należy pamiętać, że przepisywanie na bufor i z bufora - analogicznie jak to było z bębniem magnetycznym - postępuje w kierunku malejących wartości drumplace, które określają numery miejsc na buforze. Przy tym wyrazy tablic są wprowadzane w takiej kolejności, jak na bębni.

Należy również pamiętać, że do pamięci karuzelowej (o ile taka jest zainstalowana) można się dostać tylko przez bufor. Wobec tego w programach, w których przewidziana jest współpraca z karuzelą, korzystanie z pamięci buforowej jest ograniczone.

W jednym i tym samym programie można korzystać równolegle ze wszystkich rodzajów pamięci, na ile tylko pozwala zapas wolnych miejsc.

Na zakończenie warto przypomnieć, że współpraca z buforem jest około trzydziestu razy szybsza niż z bębniem.

### 5.71. Zadanie

Jak należy zmienić program z zadania 5.69, aby nie korzystać wcale z bębna a tylko z bufora? Jak duże układy równań można takim programem liczyć? Zakładamy posługiwanie się translatozem GIER ALGOL III BUF.

### 5.72. Zadanie

Zmienić program z zadania 5.69 tak, aby korzystając zarówno z pamięci buforowej, jak i bębnowej, można było nim rozwiązywać układy aż do 105 równań i tyluż niewiadomych. Zakładamy posługiwanie się translatozem GIER ALGOL III BUF.

### 5.73. Współpraca z pamięcią karuzelową w przypadku translatora GIER ALGOL III BUF

#### Procedury to car i from car

Jak już o tym była mowa, pamięć karuzelowa ma bezpośrednie połączenie jedynie z pamięcią buforową. Gdy zatem chcemy przesłać na przykład tablicę A z pamięci operacyjnej do karuzelowej, musimy tę tablicę najpierw przesłać na bufor, a dopiero z bufora na karuzelę. Analogicznie, tablicę A najpierw z karuzeli trzeba przesłać na bufor, aby później przepisać ją do pamięci operacyjnej. Wobec tego, że współpraca z buforem została już omówiona w poprzednich paragrafach, obecnie omówimy współpracę karuzeli z buforem. Współpraca ta odbywa się za pośrednictwem procedur to car i from car.

Obie procedury, tzn. to car i from car są procedurami нефункцийнymi o czterech następujących parametrach:

- nr szpuli
- nr pierwszego użytego bloku
- ilość użytych bloków
- pierwszy użyty adres na buforze

Parametry te mogą być oddzielane przecinkami albo ciągami znaków

)nazwa złożona z samych liter:(

tak jak to ma miejsce w przypadku procedur niestandardowych.

Szpule taśmy magnetycznej na karuzeli są numerowane od 0 do 63 włącznie. Zamiast numeru szpuli można użyć dowolnego wyrażenia arytmetycznego typu integer o wartości nieujemnej. Wartość tego wyrażenia jest wtedy brana modulo 64, tzn. z wartości tej uwzględnia się tylko resztę dodatnią z podzielenia przez 64, która zawsze jest liczbą całkowitą między 0 i 63.

Każda szpula zawiera 16 bloków numerowanych od 0 do 15 włącznie. Zamiast numeru pierwszego użytego bloku można w wywołaniu którejkolwiek z procedur to car i from car użyć dowolnego wyrażenia arytmetycznego typu integer o wartości nieujemnej. Wartość tego wyrażenia jest wtedy brana modulo 16, tzn. z wartości tej uwzględnia się tylko resztę dodatnią z podzielenia przez 16, która zawsze jest liczbą całkowitą między 0 i 15. W takim przypadku iloraz z powyższego dzielenia przez 16 zostaje dodany do wyrażenia arytmetycznego na numer szpuli jeszcze przed wzięciem reszty modulo 64. Jeśli na przykład wartość wyrażenia na numer szpuli będzie 126, a wartość wyrażenia na numer pierwszego bloku 71, to ponieważ  $71 = 4 \times 16 + 7$ , efektywnym numerem bloku będzie 7, a iloraz 4 zostaje dodany do 126, dając 130. Ponieważ reszta z podzielenia 130 przez 64 jest równa 2, otrzymujemy ostatecznie 2 jako efektywny numer szpuli.

Powyższy sposób wyznaczania numeru bloku i szpuli może mieć zastosowanie w programowaniu. Można mianowicie zmieniać wywoływaną szpulę, nie zmieniając jej numeru w wywołaniu procedury to car czy from car, przez powiększenie odpowiednio tylko numeru bloku. Jeżeli na przykład mieliśmy ostatnio do czytania ze szpulą nr 11 i blokiem nr 5, to wywołanie szpuli o tym samym numerze 11 ale z blokiem nr 24 da w wykonaniu szpulę nr 12 z blokiem nr 8, ponieważ  $24 = 1 \times 16 + 8$  i  $11 + 1 = 12$ .

W jednym przesyłaniu można wykorzystywać tylko jedną szpulę, a przysyłać można tylko pełnymi blokami. Jednorazowo można przysłać co najwyżej 8 bloków czyli 4096 liczb, bo taka jest pojemność bufora. Ilość przesyłanych liczb otrzymujemy mnożąc liczbę użytych bloków przez 512, ponieważ każdy blok ma pojemność 512 słów.

Z powyższego wynika, że trzeci parametr w wywoływaniu procedury to car czy from car, tzn. ilość użytych bloków, musi mieć taką całkowitą wartość nie mniejszą niż 0 i nie większą niż 8, aby dodanie jej do efektywnego numeru pierwszego użytego bloku nie przekroczyło 16, gdyż tyle jest bloków na jednej szpuli. Jeśli na przykład efektywnym numerem pierwszego użytego bloku jest 3, ilość bloków nie może być większa niż 8, jeśli efektywnym numerem pierwszego bloku jest dowolna liczba całkowita nie mniejsza niż 0 i nie większa niż 8, ilość użytych bloków nie może być również większa niż 8, jeśli na koniec numerem pierwszego użytego bloku jest 15, ilość bloków może być tylko 0 lub 1. Gdy liczba użytych bloków w tym czy również innym przypadku jest zerem, przesyłanie w ogóle nie następuje.

Zamiast pierwszego użytego adresu na buforze może być użyte dowolne wyrażenie arytmetyczne typu integer o wartości

nieujemnej. Wartość tego wyrażenia jest wtedy brana modulo 4096 (tyle jest bowiem miejsc na buforze), tzn. zostaje uwzględniona tylko dodatnia reszta z podzielenia tej wartości przez 4096. Taka reszta jest zawsze liczbą całkowitą nieujemną między 0 i 4095 włącznie.

Istnieje możliwość mechanicznego zablokowania dostępu do każdej ze szpul karuzeli. Chroni to informacje zapisane na szpuli od nieprzewidzianego a niepożądanego zniszczenia. Blokade taką w razie potrzeby należy specjalnie zamówić u operatora obsługującego maszynę.

Gospodarka wszystkimi miejscami pamięci karuzelowej należy do programującego, z wyłączeniem oczywiście szpul zablokowanych. Nie zaleca się jednak korzystania ze szpuli nr 63, ponieważ zazwyczaj używa się jej do przechowywania translatora.

Wywołanie którejkolwiek z procedur to car i from car powoduje również automatyczną kontrolę prawidłowości przesyłania i ewentualne jego powtarzanie aż do poprawnego rezultatu. Każdorazowe powtórzenie przesyłania (na skutek błędu w poprzednim przesyłaniu) jest sygnalizowane automatycznie przez wydrukowanie czerwonym drukiem na maszynie do pisania liter CA.

A oto przykłady wywołania procedur to car i from car:

- 1) to car(27,13,2,1385)
- 2) from car(57,0,4,2x+n-1)
- 3) from car(48) nr bloku:(323) ilość bloków:(6) adres bufora:(7244)

W p r z y k ł a d z i e 1 mamy do czynienia z instrukcją przepisania z bufora od adresu 1385 począwszy 1024 liczb na karuzelę (2 bloki po 512 liczb) na szpulę nr 27 na bloki nr 13 i nr 14 (pierwszy blok ma być z numerem 13, a bloków jest 2).

W p r z y k ł a d z i e 2 mamy do czynienia z instrukcją odczytania z karuzeli ze szpuli nr 57, od bloku nr 0 począwszy, 2048 liczb (4 bloki po 512 liczb) i zapisania ich na buforze od tego adresu począwszy, który jest wyznaczony aktualną wartością wyrażenia  $2 \times n + m - 1$ . Gdyby na przykład w chwili wywołania danej instrukcji było  $n=1100$  i  $m=307$ , wartością wymienionego wyrażenia byłaby liczba 2506 i zapis na buforze rozpocząłby się od adresu 2506. Ostatnia z przesłanych liczb znalazłaby się zatem na buforze pod adresem 459.

W p r z y k ł a d z i e 3 mamy do czynienia z instrukcją, w której dla oddzielenia parametrów zamiast przecinków użyto wspomnianych wyżej ciągów znaków, stanowiących komentarz dla wywoływanych parametrów. Dana instrukcja jest oczywiście równoznaczna dla maszyny z instrukcją

from car(48,323,6,7244)

Ponieważ  $321 = 20 \times 16 + 3$ , instrukcja ta jest z kolei równoznaczna z

from car(68,3,6,7244)

a ta z instrukcją

from car(4,3,6,3148)

ponieważ  $68=1 \times 64 + 4$  i  $7244=1 \times 4096 + 3148$ . Mamy zatem do czytania z instrukcją odczytania z karuzeli 3072 liczb ( $6 \times 512 = 3072$ ), mianowicie bloków nr nr 3,4,5,6,7,8 ze szpuli nr 4, i przesłania na bufor począwszy od adresu 3148 (ostatnia liczba znajdzie się na buforze pod adresem 77).

#### 5.74. Zadanie

Napisać program, za pomocą którego można umieścić w pamięci karuzelowej tablicę jednowymiarową, zawierającą kolejne liczby naturalne od 1 do 512 włącznie, jeśli wiadomo, że na buforze są wolne miejsca począwszy od adresu 3000, a tablica ma być zapisana na szpuli nr 39 w bloku nr 10. Zakładamy posługiwanie się translatorem GIER ALGOL III BUF.

#### 5.75. Zadanie

Napisać program, za pomocą którego można przepisać zawartość bloków 8,9,10 i 11 na szpuli nr 28 z karuzeli na bęben, jeżeli wiadomo, że cały bufor jest wolny. Zakładamy posługiwanie się translatorem GIER ALGOL III BUF.

#### 5.76. Procedury dla fragmentów w kodzie wewnętrznym maszyny

Translator GIER-ALGOLu zawiera pięć procedur standardowych: pack, split, gier, gierproc i gierdrum, mających za zadanie umożliwienie programującemu wplatania do programu napisanego w języku GIER-ALGOL fragmentów, napisanych w języku wewnętrznym maszyny. Ponieważ dla zrozumienia działania tych procedur konieczna jest znajomość języka wewnętrznego maszyny GIER, której tutaj nie zakładamy, poprzestaniemy na bardzo ogólnikowej charakterystyce wymienionych procedur.

**P r o c e d u r a pack** (czytaj: pak) służy do montowania słów w języku wewnętrznym maszyny, według wzorów podawanych za pośrednictwem całkowitych liczb binarnych dla dowolnych części słowa.

**P r o c e d u r a split** (czytaj: splyt) działa przeciwnie do pack: rozбивa słowo na części mające być wzorami do późniejszego działania.

**P r o c e d u r a gier** przekazuje sterowanie programem do instrukcji, zapisanej w postaci całkowitej liczby binarnej, będącej aktualną wartością zmiennej stanowiącej aktualny parametr procedury gier.

**P r o c e d u r a gierproc** umożliwia wczytanie fragmentu programu w kodzie wewnętrznym maszyny GIER z taśmy papierowej na czytniku i jednorazowe wykonanie tego fragmentu, jak



również wplatanie do fragmentów programu w kodzie wewnętrznym parametrów z części programu zapisanych w języku GIER-ALGOL, z tym, że takie parametry są parametrami aktualnymi procedury gierproc.

Procedura gierdrum umożliwia wczytanie fragmentu w kodzie wewnętrznym z taśmy papierowej na bęben magnetyczny.

#### 5.77. Współpraca z pamięcią buforową w przypadku translatora GIER ALGOL III

Translator GIER ALGOL III nie zawiera procedur standardowych to buf, from buf, to car i from car, i współpraca z buforem odbywa się w inny sposób aniżeli w przypadku posługiwania się translatorem GIER ALGOL III BUF.

Przed wszystkim przesyłać można tylko tablice. Nie można zatem przesyłać wartości zmiennych nieindeksowanych, jak to było dozwolone dla translatora GIER ALGOL III BUF. Aby było możliwe przepisywanie tablic z pamięci operacyjnej do buforowej lub odwrotnie, oprócz procedury standardowej gierproc potrzebna jest jeszcze specjalna procedura o nazwie corbuf (czytaj: korbuf) napisana w kodzie wewnętrznym maszyny i przechowywana stale jako tablica jednowymiarowa o 16 wyrazach na taśmie papierowej u operatora maszyny.

Jeśli w programie przewidujemy korzystanie z pamięci buforowej, należy przede wszystkim wprowadzić procedurę corbuf z taśmy papierowej na bęben magnetyczny. Wykorzystuje się do tego procedurę standardową gierdrum w sposób następujący:

W nagłówku bloku obejmującego wszystkie miejsca programu, w których będziemy korzystać z pamięci buforowej, deklarujemy dwie zmienne typu integer, na przykład zmienne c i k. Wewnątrz tego bloku, ale przed pierwszym miejscem, w którym będzie wykorzystana pamięć buforowa, umieszczamy dwie instrukcje:

```
c:= drumplace;
```

```
gierdrum(<corbuf>,k);
```

Pierwsza z nich spowoduje zapamiętanie aktualnej wartości drumplace do późniejszego przepisania procedury corbuf z bębna do pamięci operacyjnej. Druga spowoduje odczytanie procedury corbuf z taśmy papierowej, którą uprzednio - po wzięciu od operatora - musimy założyć na czytnik, i napisanie jej na bębnie, wraz z podstawieniem na k liczby miejsc zajętych przez procedurę corbuf na bębnie (dla używanego ostatnio programu corbuf będzie tu podstawiona wartość k:= 16). Po wykonaniu powyższych dwu instrukcji wartością drumplace jest c-k.

Z procedury corbuf korzystamy za każdym razem, gdy przepisujemy jakiś materiał z pamięci operacyjnej do buforowej lub odwrotnie. W tym momencie tablica z procedurą corbuf musi znajdować się w pamięci operacyjnej, a zatem musi być zadeklarowana w bloku obejmującym dane miejsce programu jako

```
array corbuf[1:k]
```

Wynika stąd, że blok, w nagłówku którego została zadeklarowana zmienna *k*, musi być zewnętrznym w stosunku do tego, w nagłówku którego deklarujemy *corbuf*, przy czym otwarcie tego drugiego bloku musi następować już po wspomnianej instrukcji *gierdrum*, tzn. gdy została już przez tę procedurę podstawiona wartość na zmienną *k*.

Wewnątrz bloku z deklaracją *corbuf* sprowadzenie tej procedury z bębna zostaje wykonane poprzez instrukcje:

```
drumplace:= c;
```

```
from drum(corbuf);
```

Powstaje przez to sytuacja, w której komunikacja między pamięcią operacyjną i buforową jest już możliwa. Przepisywanie tablicy dokonuje się przez instrukcję

```
gierproc(corbuf[2], nazwa tablicy, pierwszy adres w pamięci
buforowej, od którego począwszy dana tablica jest zapisywana,
wyrażenie booleowskie)
```

Jeśli w tej instrukcji wyrażenie booleowskie ma wartość true, odbywa się przepisywanie danej tablicy z pamięci operacyjnej do buforowej, jeśli false, to odwrotnie.

Zamiast pierwszego adresu w pamięci buforowej, można podać wyrażenie arytmetyczne typu integer, którego wartość określa pierwszy adres. Wartość tego wyrażenia jest brana modulo 4096, tzn. jest brana zamiast niej reszta dodatnia z podzielenia danej wartości przez 4096. W ten sposób wartość danego wyrażenia jest zawsze sprowadzana do liczb całkowitych od 0 do 4095 włącznie, a więc do numerów czyli adresów miejsc w pamięci buforowej.

Procedura *gierproc* należy do pseudofunkcyjnych, ale jej wartość po przepisaniu tablicy z pamięci operacyjnej do buforowej lub odwrotnie jest nonsensowna.

Procedura *corbuf* nie przeprowadza żadnej kontroli poprawności transportu materiału między pamięcią operacyjną a buforową.

Opisany wyżej sposób współpracy z buforem można stosować także w przypadku translatora GIER ALGOL III BUF, ale sposób ten jest wtedy nieopłacalny wobec znacznie wygodniejszej możliwości korzystania z procedur standardowych *to buf* i *from buf*.

## 5.78. Zadanie

Tablicę jednowymiarową, zawierającą liczby naturalne od 1000 do 1499 włącznie należy zapisać na buforze począwszy od adresu 4000. Zakładamy współpracę z buforem poprzez translator GIER ALGOL III.

Jak wyglądałby taki program przy translatorze GIER ALGOL III BUF?



### 5.79. Współpraca z pamięcią karuzelową w przypadku translatora GIER ALGOL III

Translator GIER ALGOL III nie zawiera procedur standardowych to buf, from buf, to car i from car, i współpraca z karuzelą odbywa się w inny sposób aniżeli w przypadku posługiwania się translatorem GIER ALGOL III BUF.

Ogólne zasady są podobne. Pamięć karuzelowa ma bezpośrednie połączenie tylko z pamięcią buforową. Przesyłania mogą zachodzić tylko w granicach jednej szpuli taśmy magnetycznej, tylko pełnymi blokami po 512 liczb każdy, ograniczenia na ilość jednorazowo przesyłanych bloków są takie same, te same cztery parametry są potrzebne do wywołania instrukcji przesłania, a mianowicie nr szpuli, nr pierwszego użytego bloku, ilość użytych bloków i pierwszy adres użyty na buforze. W ten sam sposób można na powyższe cztery parametry podstawiać wyrażenia arytmetyczne typu integer i w taki sam sposób otrzymuje się efektywne wartości tych czterech parametrów.

Aby jednak było możliwe przepisanie materiału z karuzeli na bufor lub odwrotnie, musimy mieć w pamięci operacyjnej tablicę jednowymiarową o 28 wyrazach, zawierającą specjalną procedurę pod nazwą bufcar (czytaj: bafkar). Procedurę tę wprowadzamy tak jak corbuf z taśmy papierowej za pomocą procedury gierdrum na bęben magnetyczny. Jeśli chcemy korzystać z pamięci karuzelowej, to ze względu na konieczność przechodzenia przez pamięć buforową, z reguły wprowadzamy procedury corbuf i bufcar łącznie. W tym celu w nagłówku bloku obejmującego wszystkie miejsca programu, w których będziemy używać pamięci buforowej lub karuzelowej, deklarujemy trzy zmienne typu integer, np. zmienne c, k, b. Wewnątrz tego bloku, ale przed pierwszym miejscem, gdzie będzie wykorzystana pamięć buforowa lub karuzelowa, umieszczamy instrukcje

```
c:= drumplace;
gierdrum(<<corbuf>>,k);
gierdrum(<<bufcar>>,b);
```

Pierwsza z nich powoduje zapamiętanie aktualnej wartości drumplace do późniejszego przepisania procedur corbuf i bufcar z bębna do pamięci operacyjnej. Dwie następne powodują odczytanie procedur corbuf i bufcar z taśmy papierowej, którą uprzednio – po wzięciu od operatora – musimy założyć na czytelnik przestrzegając kolejności tych procedur, i zapisanie ich w postaci tablic na bębnie, wraz z podstawieniem na k i b liczby miejsc zajętych odpowiednio przez corbuf i bufcar na bębnie (dla ostatnio używanych programów zostają tu podstawione wartości k:= 16 i b:= 28). Po wykonaniu powyższych instrukcji wartością drumplace jest c-k-b.

Aby można było z procedur corbuf i bufcar korzystać, muszą być one w odpowiednich momentach sprowadzane z bębna do pamięci operacyjnej, co wymaga uprzedniej deklaracji:

```
array corbuf[1:k], bufcar[1:b]
```

Blok, w którego nagłówku muszą być umieszczone powyższe deklaracje, musi leżeć wewnątrz bloku, w którego nagłówku były deklarowane zmienne *k i b*, i to już po wspomnianych instrukcjach *gierdrum*, tzn. gdy zostały już przez tę procedurę podstawione wartości *k i b*.

Wewnątrz bloku z deklaracjami *corbuf* i *bufcar* sprowadzenie tablic z tymi procedurami z bębna do pamięci operacyjnej zostaje wykonane poprzez instrukcje:

```
drumplace:= c;
from drum(corbuf);
from drum(bufcar);
```

Powstaje przez to sytuacja, w której komunikacja między pamięcią operacyjną i buforową, jak również między pamięcią buforową i karuzelową jest już możliwa. Przepisywanie następuje przez instrukcję:

```
gierproc(bufcar[2], nr szpuli, nr pierwszego wykorzystywanego
bloku, liczba wykorzystywanych bloków, pierwszy wykorzystywa-
ny adres na buforze, wyrażenie booleowskie)
```

Jeśli w tej instrukcji wyrażenie booleowskie ma wartość true, przepisywanie odbywa się z pamięci buforowej na karuzelową, jeśli false, to odwrotnie.

Jeśli transport z pamięci buforowej do karuzelowej przebiegł prawidłowo, wartość *gierproc* staje się z chwilą ukończenia transportu równa 0. Jeśli transport dwukrotnie przebiegł wadliwie, nie zostaje w ogóle wykonany, a *gierproc* otrzymuje wartość równą 1. Fakt ten wykorzystuje się w celach kontrolnych.

Opisany wyżej sposób współpracy z karuzelą można stosować także w przypadku translatora GIER ALGOL III BUF, ale sposób ten jest wtedy nieopłacalny wobec znacznie wygodniejszej możliwości korzystania z procedur standardowych to *car* i *from car*.

### 5.80. Zadanie

Rozwiązać zadanie 5.75 przy założeniu posługiwania się translatozem GIER ALGOL III.