ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1980

Seria: AUTOMATYKA z. 53 Nr kol. 649

Jan BRUSKI, Bogdan GRUDZIŃSKI

SYSTEM PROCEDUR DO PRZETWARZANIA LIST STOSOWYCH W JĘZYKU ALGOL 1900

Część 2. Procedury pomocnicze. Cechy użytkowe i zastosowania

Streszczenie. Artykuł stanowi kontynuację opisu systemu procedur służących do przetwarzania list stosowych w programach opracowanych w języku ALGOL 1900. Zawiera on krótki opis pomocniczych procedur systemu oraz sposób korzystania z nich. W artykule przedstawiono również charakterystyczne cechy systemu oraz pewne możliwości jego stosowania.

System procedur służący do tworzenia i wykorzystywania list stosowych w języku ALGOL 1900 składa się z dwóch pakietów procedur: podstawowych i pomocniczych. Procedury podstawowe zostały opisane w pierwszej części opracowania, natomiast teraz przedstawiony zostanie opis procedur pomocniczych.

Jak podano wcześniej, elementy list mają strukturę dwukomórkową. Do zapisu wartości przeznaczona jest jedna komórka. Głowy list, poprzez które możliwy jest dostęp do elementów list, opisuje się w programie jako dwuelementowe tablice typu INTEGER. Wynika stąd, że wartościami elementów list mogą być jedynie liczby całkowite. Nie przeszkadza to jednak w interpretowaniu wartości elementów list inaczej, np. jako części liczb rzeczywistych, wartości logicznych lub danych upakowanych. Swobodę interpretacji ułatwiają pomocnicze procedury systemu, których zadania zostaną niżej krótko omówione.

Procedura BOININT jest funkcyjną, jednoparametrową procedurą typu INTEGER. Parametrem jej jest wyrażenie logiczne. Procedura ta dokonuje formalnej zmiany typu wyrażenia z logicznego na arytmetyczne całkowite. Jeżeli wartość parametru aktualnego procedury jest równa TRUE, wartością procedury jest liczba całkowita 1, w przeciwnym razie - liczba całkowita Ø. Procedurę BOININT wykorzystuje się, gdy wartością elementu listy ma być wartość logiczna. Przykładowo, chcąc umieścić w liście LX wartość iloczynu logicznego dwóch zmiennych boolowskich, trzeba użyć następującej sekwencji instrukcji:

```
LX [Ø] : = BOININT (A ∧ B) ;
PRESERVE (LX [Ø] );
```

Dla umożliwienia zapisu wartości rzeczywistych w elementach list stosuje się trójparametrową procedurę REININT. Służy ona do rozpakowania dwukomórkowych wartości typu REAL na dwie wartości typu INTEGER. Pierwszym parametrem procedury jest wyrażenie arytmetyczne typu rzeczywistego, dwa następne są zmiennymi całkowitymi, do których następuje rozpakowanie wartości pierwszego parametru.

Chcąc przykładowo umieścić w liście LX wartość wyrażenia sin x + + x cos x, należy użyć następującej sekwencji instrukcji:

```
REININT (SIN(X) + X x COS(X), A, B);
LX [Ø] : = B ;
PRESERVE (LX [Ø]) ;
LX [Ø] : = A ;
PRESERVE (LX [Ø]) ;
```

Należy zauważyć, że w wyniku wykonania tych instrukcji, w pierwszym elemencie ogona listy LX umieszczona zostanie pierwsza część wartości wyrażenia arytmetycznego, w drugim elemencie ogona - część druga. Kolejność ta mogłaby być inna, lecz przy odtwarzaniu wartości wyrażenia na podstawie dwóch zapamiętanych liczb całkowitych, istotna jest znajomość tej kolejności.

Do odtwarzania pierwotnych wartości umieszczonych w liście, służą procedury INTINBO i INTINRE.

Procedura INTINBO jest funkcyjną, jednoparametrową procedurą typu BOOLEAN. Jej wartością jest TRUE, gdy wartością parametru aktualnego jest liczba 1 oraz FALSE, gdy wartość parametru jest równa Ø. Procedura ta dokonuje więc formalnej zamiany typów wartości: z całkowitego na boolowski.

Odtworzenie wartości logicznej umieszczonej w pierwszym elemencie ogona listy LX wymaga użycia sekwencji instrukcji:

```
RENOVATE (LX [Ø] , ETYK);
ZB : = INTINBO (LX [Ø]);
```

gdzie ZB jest zmienną typu BOOLEAN.

Procedura INTINRE jest dwuparametrową, funkcyjną procedurą typu REAL. Jej wartością jest liczba rzeczywista uzyskana przez formalne upakowanie dwóch liczb całkowitych, będących jej parametrami.

Odtworzenie wartości wyrażenia arytmetycznego, umieszczonej w postaci dwóch liczb całkowitych w pierwszym i drugim elemencie ogona listy LX, wymaga użycia poniższej sekwencji instrukcji:

```
RENOVATE (LX [Ø], ETYK);
A : = LX [Ø];
RENOVATE (LX [Ø], ETYK);
B : = LX[Ø];
ZR: = INTINRE (A, B);
```

gdzie ZR jest zmienną typu REAL.

Należy zwrócić uwagę na kolejność parametrów A i B w instrukcji procedury INTINRE (A, B). Musi ona być zgodna z kolejnością, jaką zastosowano podczas rozpakowywania wartości typu REAL i zapisu do elementów listy.

System przetwarzania list stosowych umożliwia użytkownikowi oznaczanie wartości przechowywanych w listach pewnymi znacznikami. Nie jest to konieczne ale może ułatwić kontrolę nad programem, szczególnie w fazie jego uruchamiania i testowania.

W celu zaznaczenia, że wartości elementów listy winny być traktowane jako wartości o typach INTEGER, BOOLEAN lub REAL, do części adresowych tych elementów można zapisać odpowiednie znaczniki.
W przypadku wartości typu REAL wprowadzono dwa znaczniki: inny dla pierwszego słowa wartości rzeczywistej, inny dla drugiego słowa. Zapisu znaczników do części adresowej głowy listy dokonuje się przez użycie procedur: SETI, SETB, SETR1 i SETR2. Jedynym parametrem każdej z tych procedur jest zmienna ze wskaźnikiem Ø, reprezentująca nazwę listy.

Umieszczenie w ogonie listy LX wartości wyrażenia boolowskiego i oznaczenie tej wartości znacznikiem typu BOOLEAN wymaga użycia następującej sekwencji instrukcji:

```
LX [Ø] : = BOINT (A ∧ B ∨ C = D);
SETB (LX [Ø] );
PRESERVE (LX [Ø]);
```

W wypadku gdy wartością elementu ogona listy ma być liczba całkowita, a wartość ta ma być oznaczona znacznikiem typu INTEGER, trzeba umieścić w programie poniższą sekwencję instrukcji:

```
LX [Ø] : = I ÷ J + 12;
SETI (LX [Ø]);
PRESERVE (LX [Ø]);
```

Umieszczenie w liście LX wartości zmiennej rzeczywistej X i oznaczenie pierwszej części tej wartości znacznikiem typu REAL-1 a drugiej części znacznikiem REAL-2 może się odbyć następująco:

```
REININT (X, A, B);
LX [Ø] : = B;
SETR2 (LX [Ø]);
PRESERVE (LX [Ø]);
LX [Ø] : = A;
SETR1 (LX [Ø]);
PRESERVE (LX [Ø]);
```

Po zapisie dowolnego znacznika do części adresowej głowy listy, znacznik ten będzie tam przechowywany aż do ponownego użycia jednej z procedur SETI, SETB, SETR1, SETR2. Każde użycie procedury PRESERVE wywołuje powielenie znacznika do części adresowej pierwszego elementu ogona listy. Dzięki temu wszystkie kolejne wartości zapisane w liście będą oznaczone tym samym znacznikiem.

Testowanie typów wartości przechowywanych w elementach list odbywa się przez użycie jednoparametrowych, funkcyjnych procedur typu BOOLEAN. Są to procedury INT, BOOL, REAL1, REAL2.
Parametrem każdej z nich jest zmienna ze wskaźnikiem Ø reprezentująca nazwę listy. Procedury te przyjmują wartość TRUE, gdy parametr aktualny zawiera znacznik odpowiedniego typu i wartość FALSE w przeciwnym przypadku.

Przykładowo chcąc zbadać czy pierwszy element ogona listy zawiera pierwsze słowo wartości typu REAL (tzn. znacznik typu REAL-1), należy zastosować instrukcje:

```
RENOVATE (LX [Ø], ETYK);
IF REAL1 (LX [Ø]) THEN GO TO E;
```

Przeniesienie sterowania do etykiety E nastąpi wtedy, gdy część adresowa głowy listy LX zawiera znacznik typu REAL-1.

Istnieje jeszcze jedna możliwa interpretacja wartości elementów list. Jest nią traktowanie tych wartości jako danych upakowanych. Możliwość tę stwarza grupa pięciu procedur: PACK, UNPACK, FIXFLOAT, FLOATFIX oraz MASK.

Procedura PACK jest czteroparametrową procedurą, służącą do umieszczania w części komórki pamięci nieujemnej liczby całkowitej. Pierwszy parametr stanowi nazwę miejsca, do którego ma nastąpić zapis (zmienna ze wskaźnikiem Ø reprezentująca nazwę listy), drugi parametr określa numer bitu, od którego zaczyna się pole przeznaczone dla zapisywanej liczby, parametr trzeci określa długość tego pola (liczbę bitów), natomiast czwartym i ostatnim parametrem jest wartość, która ma być zapisana w danym polu. Para-

metry drugi, trzeci i czwarty mogą być wyrażeniami arytmetycznymi typu INTEGER.

Przykładowo, zapis liczby 17 do głowy listy LX na 5 bitach, począwszy od bitu 7, wymaga umieszczenia w programie instrukcji procedury:

```
PACK (LX [Ø] , 7, 5, 17);
```

W wyniku wykonania tej instrukcji na bitach B_7, B_8, B_9, B_{10}, B_{11} zostanie umieszczona liczba 17, natomiast pozostałe bity zmiennej LX [Ø] pozostaną nienaruszone. Bit B_0 jest najbardziej znaczącym bitem słowa, bit B_{23} - najmniej znaczącym.

Procedura UNPACK ma działanie odwrotne. Służy ona do odczytu nieujemnej liczby całkowitej umieszczonej w części komórki pamięci. Posiada ona cztery parametry. Pierwszy parametr jest nazwą miejsca (zmienną ze wskaźnikiem Ø reprezentującą nazwę listy), z którego trzeba dokonać odczytu. Parametry drugi i trzeci mają to samo znaczenie, co w przypadku procedury PACK, natomiast czwarty parametr powinien być zmienną typu całkowitego. Do zmiennej tej wpisuje się liczbę otrzymaną w wyniku rozpakowania odpowiedniej części komórki pamięci.

Odzyskanie upakowanej w poprzednim przykładzie liczby 17 i umieszczenie jej w zmiennej B nastąpi w wyniku wykonania instrukcji procedury

```
UNPACK (LX[Ø] ,7, 5, B);
```

Elementów list można używać również do przechowywania upakowanych liczb rzeczywistych z pewnego ograniczonego zakresu. Możliwość tę uzyskano dzięki zamianie liczb rzeczywistych w postaci zmiennoprzecinkowej (półlogarytmicznej) do postaci stałoprzecinkowej ze sterowanym ustawianiem kropki binarnej, oddzielającej część całkowitą liczby od ułamkowej.

Zamiany tej dokonuje dwuparametrowa, funkcyjna procedura typu INTEGER.FIXFLOAT. Pierwszym parametrem tej procedury jest wyrażenie arytmetyczne typu rzeczywistego, którego wartość zostanie upakowana w postaci jednokomórkowej, drugi natomiast parametr określa ilość cyfr binarnych części ułamkowej zamienionej liczby, a więc dokładność zapamiętania liczby. Wartością funkcji FIXFLOAT jest liczba całkowita, będąca obrazem zamienionej do postaci stałoprzecinkowej liczby rzeczywistej.

W celu zapamiętania np. liczby 5.3 z dokładnością 6 cyfr binarnych po kropce, wystarczy w programie umieścić instrukcję:

```
LX [Ø] : = FIXFLOAT (5.3, 6);
```

Odwrotną zamianę przeprowadza procedura FLOATFIX. Jest to dwuparametrowa procedura funkcyjna typu REAL. Dokonuje ona zamiany liczby całkowitej będącej obrazem liczby rzeczywistej w postaci stałoprzecinkowej, do postaci zmiennoprzecinkowej. Pierwszy parametr procedury jest wyrażeniem arytmetycznym typu całkowitego. Wartość tego wyrażenia jest traktowana jako liczba rzeczywista w postaci stałoprzecinkowej, której liczba cyfr po krop-

ce binarnej określona jest przez drugi parametr procedury. Wartością funkcji FLOATFIX jest liczba rzeczywista w postaci zmiennoprzecinkowej, uzyskana w wyniku zamiany wartości pierwszego parametru.

Po to, by odtworzyć wartość rzeczywistą 5.3 z poprzedniego przykładu, należy zastosować instrukcję:

```
R : = FLOATFIX (LX [Ø], 6);
```

Operując upakowanymi danymi, czasami jest wygodne korzystanie z pewnej dodatkowej procedury funkcyjnej typu INTEGER o nazwie MASK. Służy ona do maskowania (wykonywania iloczynu logicznego pozycyjnego) liczby całkowitej z przesunięciem wyniku o określoną liczbę bitów. Maskę (drugi argument iloczynu logicznego) stanowią reprezentacje binarne czterech pierwszych parametrów procedury. Parametry te są wyrażeniami arytmetycznymi typu całkowitego o wartościach nieujemnych, nie przekraczających liczby 63. Liczba bitów maski jest równa 24, a więc odpowiada długości słowa maszyny cyfrowej. Piąty parametr procedury jest wyrażeniem arytmetycznym typu całkowitego, którego wartość podlega maskowaniu, natomiast szósty parametr określa liczbę przesunięć wyniku w prawo.
Wartością procedury MASK jest liczba całkowita otrzymana po "wycięciu" dowolnej grupy bitów z liczby całkowitej i przesunięciu wyniku o określoną liczbę pozycji w prawo.

Przykładowo instrukcja: Q : = MASK (Ø, Ø, 63, Ø, LX[Ø], 6); powoduje:
- utworzenie maski równej 000000000000111111000000,
- koniunkcję maski i wartości zmiennej LX[Ø],
- przesunięcie wyniku o 6 bitów w prawo,
- podstawienie tak uzyskanej liczby pod zmienną Q.

Należy zwrócić uwagę, że procedury związane z pakowaniem i rozpakowaniem danych mają charakter ogólny i można je stosować także wtedy, gdy nie używa się list stosowych.

Trzy z omówionych wyżej procedur, a mianowicie: PACK, UNPACK i MASK, dodatkowo wykonane są w innej nieco wersji, w której są nazwane: TPACK, TUNPACK i TMASK. Oprócz swych podstawowych zadań opisanych wyżej, sprawdzają one poprawność parametrów i sygnalizują ewentualne błędy. Procedury te stosuje się głównie w trakcie uruchamiania i testowania programów.

System procedur do przetwarzania list stosowych powstał, jako system pilotujący dla systemu przetwarzania dowolnych struktur danych. Jego główne przeznaczenie stanowi zwiększenie możliwości i elastyczności języka ALGOL 1900. Walory tego systemu będzie oczywiście można w pełni ocenić dopiero po zdobyciu doświadczeń eksploatacyjnych, po pewnym okresie jego wykorzystywania. Jednak już w tej chwili można wyeksponować pewne cechy tego systemu, które pozwalają przynajmniej na cząstkową jego ocenę.

Główną zaletą systemu jest umożliwienie stosowania w programach napisanych w języku ALGOL 1900, liniowych obiektów o niezdeterminowanych z góry

wielkościach. Obiekty te zbudowane są jako listy stosowe ale można je traktować jako jednowymiarowe tablice typu całkowitego o nieokreślonych rozmiarach.
Tego rodzaju interpretacja jest możliwa dzięki istnieniu procedur GIVE i STORE, które pozwalają na dostęp do dowolnych elementów list.

Stosując pewne odstępstwo od opisanej zasady tworzenia głów list i wykorzystując jako głowy, miejsca tablic wieloelementowych, a nawet wielowymiarowych, można budować obiekty zwane multipletami o giętkich parach granicznych, podobnie jak w języku ALGOL 68.

Dzięki zastosowaniu procedur zmiany typów wartości, stworzona została znaczna swoboda interpretacji zawartości poszczególnych miejsc pamięci. Można np. arytmetyczne wartości całkowite traktować jako wartości logiczne i wykonywać na nich operacje logiczne. Otrzymywane wtedy rezultaty stanowią wyniki logicznych działań pozycyjnych wykonywanych na wszystkich bitach argumentów.

Dzięki istnieniu procedur upakowujących i rozpakowujących, stworzona została w języku ALGOL 1900, nie istniejąca dotąd bezpośrednia możliwość wykonywania operacji na częściach słów, w tym również na pojedynczych bitach, podobnie jak np. w językach PASCAL czy ALGOL 68.

Ułatwiona została możliwość tworzenia "długich" słów (jak w ALGOL-u 68), co jest szczególnie pożyteczne w przypadku obliczeń ze zwiększoną precyzją.

Stworzona możliwość upakowania kilku liczb całkowitych lub liczby rzeczywistej w jednej komórce pamięci, pozwala często na duże oszczędności wymaganego obszaru pamięci. Może to być szczególnie korzystne w trakcie tzw. przetwarzania danych, gdyż wtedy rzędy liczb zwykle niewiele się między sobą różnią, natomiast znaczna jest liczba danych.

Cecha ta rekompensuje również częściowo straty pamięci wynikłe ze stosowania list stosowych w programach (każdy element listy składa się z dwóch słów pamięci). Przy okazji można jednak zauważyć, że wykorzystanie pamięci w programach ALGOL-u, w przypadku używania list stosowych jest porównywalne z wykorzystaniem pamięci w programach FORTRAN-owskich.

Wszystkie procedury systemu zostały napisane w języku PLAN. Pozwoliło to na ich optymalizację ze względów czasowych, a dzięki temu uzyskano szybkość wykonywania tych procedur porównywalną z szybkością wykonywania operacji zmiennoprzecinkowych. W wyniku tego spadek efektywności przetwarzania maszyn cyfrowych ODRA serii 1300 w przypadku wykorzystywania list stosowych jest niewielki i w pełni zrekompensowany zaletami systemu.

Opracowany system procedur może znaleźć bardzo różnorodne zastosowania. Niektóre z nich wynikają bezpośrednio z możliwości programowych stworzonych przez system i nie muszą być związane z jakimiś konkretnymi zagadnieniami, a raczej mają charakter ogólny i wywodzą się z potrzeby przetwarzania bloków danych. Inne natomiast wynikają z istnienia algorytmów przetwarzania list stosowych w różnych zagadnieniach. Można tu przykładowo wymie-

Tabela 1

Zestawienie procedur do przetwarzania list stosowych

Funkcje grupy procedur	Nazwa procedury	Typ procedury	Parametry	Treść procedury
1	2	3	4	5
Generowanie warunków początkowych	GENERATE	niefunkcyjna	Brak	Generowanie wewnętrznych warunków początkowych, w tym listy nieużytków
	NIL	INTEGER	Brak	Wpisywanie symbolu NIL do części adresowej głowy listy.
Tworzenie	PRESERVE	niefunkcyjna	I	Przesuwanie głowy listy do pierwszego elementu ogona bez zmiany wartości głowy listy.
Likwidacja	RENOVATE	niefunkcyjna	I,E	Przepisanie wartości z pierwszego elementu ogona do głowy listy z usunięciem przepisanego elementu z ogona listy.
Odczytywanie wartości i dokonywanie zmian w listach	LOSE	niefunkcyjna	I,E	Usunięcie z listy pierwszego elementu ogona.
	GIVE	INTEGER	I,E,I	Odczyt wartości dowolnego elementu ogona listy.
	STORE	INTEGER	I,E,I,I	Zapis wartości do dowolnego elementu ogona.
	NULL	BOOLEAN	I	Sprawdzenie czy ogon listy jest pusty.
Gospodarka pamięcią	SAVE	niefunkcyjna	Brak	"Ratowanie" obszarów pamięci, w których znajdują się elementy aktywnych list, a którym został cofnięty przydział pamięci po zamknięciu bloku.
Zmiana typu wyrażenia na typ INTEGER	BOININT	INTEGER	B	Formalna zmiana typu wyrażenia z log. na całk.
	REININT	niefunkcyjna	R,I,I	Rozpakowanie wartości wyrażenia rzeczywistego na dwie zmienne całkowite.
Zmiana typu wyrażenia z typu INTEGER	INTINBO	BOOLEAN	I	Formalna zmiana typu wyrażenia z całk. na log.
	INTINRE	REAL	I,I	Zamiana dwóch wartości typu całkowitego w wartość typu rzeczywistego.
Zapis znacznika typu	SETI	niefunkcyjna	I	Zapisanie znacznika typu INTEGER do części adresowej głowy listy.

od. tabeli 1

1	2	3	4	5
Zapis znacznika typu	SETB	niefunkcyjna	I	Zapisanie znacznika typu BOOLEAN
	SETR1	niefunkcyjna	I	Zapisanie znacznika typu REAL-1
	SETR2	niefunkcyjna	I	Zapisanie znacznika typu REAL-2
Testowanie znacznika typu	INT	BOOLEAN	I	Testowanie znacznika typu INTEGER
	BOOL	BOOLEAN	I	Testowanie znacznika typu BOOLEAN
	REAL1	BOOLEAN	I	Testowanie znacznika typu REAL-1
	REAL2	BOOLEAN	I	Testowanie znacznika typu REAL-2
Manipulacja danymi upakowanymi	PACK	niefunkcyjna	I,I,I,I	Pakowanie nieujemnej liczby całkowitej na określone bity głowy listy.
	UNPACK	niefunkcyjna	I,I,I,I	Rozpakowanie nieujemnej liczby całkowitej z określonych bitów głowy listy.
	FIXFLOAT	INTEGER	R,I	Zmiana liczby rzeczywistej na postać stałoprzecinkową ze sterowaną pozycją kropki.
	FLOATFIX	REAL	I,I	Zmiana liczby stałoprzecinkowej na rzeczywistą. Pozycja kropki binarnej jest sterowana.
	MASK	INTEGER	I,I,I,I,I,I	Wykonywanie iloczynu logicznego pozycyjnego głowy listy z dowolną maską 24-bitową, z przesuwaniem wyniku w prawo

Oznaczenia typu parametrów: I - INTEGER, R - REAL, B - BOOLEAN, E - LABEL.

nić problemy masowej obsługi, badania operacyjne, modelowanie procesów współbieżnych lub procesów przebiegających w czasie rzeczywistym.

Oddzielną, bardzo poważną klasę zagadnień stanowi modelowanie automatów abstrakcyjnych oraz modelowanie procesów związanych z translacją języków programowania.

LITERATURA

[1] Dokumentacja techniczno-ruchowa mc serii ODRA 1300. Algol - procedury pomocnicze.

[2] Foster J.M.: Przetwarzanie struktur listowych, PWN, Warszawa 1976.

[3] Rosen S. (ed.): Programming Systems and Languages, McGraw-Hill Book Company, New York 1967.

[4] Berztiss A.T.: Data Structures. Theory and Practice..Academic Press, New York 1975.

[5] Bruski J., Grudziński B.: System procedur do przetwarzania list stosowych w języku Algol 1900, część 1: Opis systemu i procedury podstawowe.

СИСТЕМА ПРОЦЕДУР ДЛЯ ОБРАБОТКИ СТОЛБОВЫХ СПИСКОВ
В ПРОГРАММАХ НА ЯЗЫКЕ АЛГОЛ 1900

Часть 2. Подсобные процедуры. Свойства и применение

Р е з ю м е

Статья является продолжением описания системы процедур служащих обработке столбовых списков в программах разработаных на языке Алгол 1900. В этой статье даётся короткое описание подсобных процедур системы и способ пользования этими процедурами. В статье предоставлены характеристические свойства и некоторые возможности применения системы.

THE SYSTEM OF PROCEDURES FOR PUSHDOWN LISTS
PROCESSING IN ALGOL 1900

Part. 2. Auxiliary Procedures. Peculiarities in the Use and Applications

S u m m a r y

The paper is a continuation of description of the system of procedures for pushdown lists processing which are applied in programs written in Algol 1900. A short description of the auxiliary procedures of the system and a manner of its use is comprised. The characteristic features of the system and some possibilities of its application are described.