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 1. Opis systemu i procedury podstawowe

Streszczenie. Artykuł zawiera opis opracowanego systemu procedur, służącego do przetwarzania list stosowych w języku ALGOL 1900. Przedstawiono ogólną koncepcję systemu, strukturę przetwarzanych obiektów oraz opis podstawowych procedur umożliwiających tworzenie list stosowych oraz korzystanie z tych list.

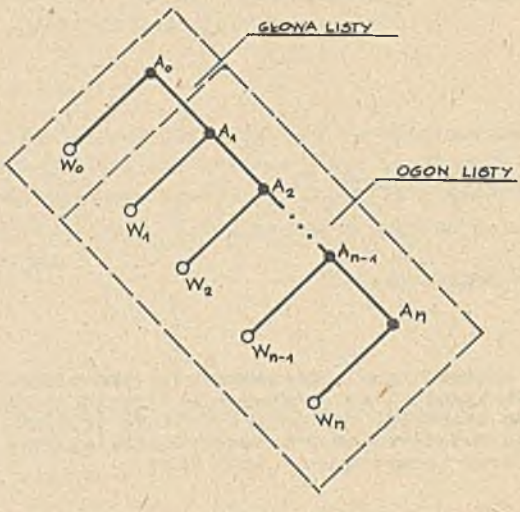
Wykorzystując elektroniczne maszyny cyfrowe do rozwiązywania zagadnień technicznych oraz naukowo-badawczych, do programowania tych zagadnień stosuje się najczęściej języki algorytmiczne takie jak: ALGOL 60, FORTRAN, PASCAL czy PL/I.
Zezwalają one na używanie zmiennych prostych oraz regularnych tablic danych, a dwa ostatnie wymienione języki także na używanie nieregularnych struktur danych. Ostatnia możliwość jest bardzo ważna, gdyż pozwala na tworzenie stosów oraz list przydatnych w procesach modelowania translatorów, a także na budowę tablic danych o nieokreślonej z góry wielkości, modelowanie grafów, tworzenie dowolnie długich łańcuchów symboli itd.
Z tego powodu zdecydowano się na budowę systemu procedur, który może być dołączony do dowolnego programu w języku ALGOL 1900.

W pierwszym etapie prac zbudowano system procedur, umożliwiających stosowanie w programach napisanych w języku ALGOL i przeznaczonych do wykonywania na maszynach cyfrowych ODRA serii 1300, obiektów zwanych listami stosowymi.

Listą stosową nazwano liniową strukturę danych, której wielkość nie jest z góry określona i która umożliwia dostęp do poszczególnych danych tylko przez swój pierwszy element zwany głową listy. Wszystkie pozostałe elementy listy stosowej (poza głową listy) określa się wspólnym mianem - ogon listy. Obrazem listy stosowej może być graf przedstawiony na rys. 1.

Zapisu informacji dokonuje się zawsze do głowy listy stosowej i przez to zwiększa się objętość listy. Odczytuje się informację również zawsze w odniesieniu do głowy listy.

Lista stosowa stanowi więc pewną realizację zespołu przechowywania danych zwanego stosem. Nie wymaga się przy tym zwartości miejsc (używania kolejnych miejsc) w ośrodku przechowywania danych. Kolejne elementy listy



Rys. 1. Graf stanowiący obraz listy stosowej

stosowej mogą w tym ośrodku zajmować zupełnie dowolne miejsca. Na podstawie zawartości dowolnego elementu listy można zawsze określić miejsce przechowywania następnego elementu (o ile ten element istnieje) lub stwierdzić, że jest to ostatni element listy stosowej. Jest to możliwe przy założeniu, że wszystkim miejscom ośrodka przechowywania danych przyporządkowane są adresy (numery) w taki sposób, że każde dwa dowolne miejsca tego ośrodka mają przyporządkowane dwa różne adresy.

Realizacja list stosowych w programach napisanych w języku ALGOL 1900 przewiduje dwukomórkową strukturę elementów tych list. Przedstawia to rys. 2.

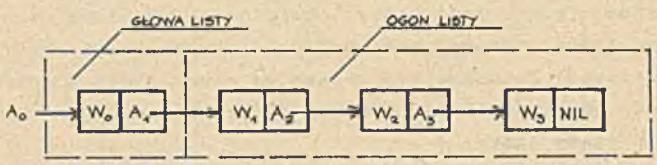
W	A

Rys. 2. Struktura dowolnego elementu listy stosowej w systemie ODRA 1300

Każdy element listy stosowej zawiera dwie informacje: wartość danej W oraz adres A następnego elementu listy.
Jeżeli rozpatrywany element listy stosowej jest ostatnim elementem tej listy, to miejsce adresu zajmuje specjalny symbol NIL.

W miejsce A oprócz adresu następnego elementu listy, może się znajdować również pewna dodatkowa informacja, której znaczenie zostanie opisane w drugiej części opracowania.

Dostęp do listy stosowej uzyskuje się poprzez podanie nazwy listy. Jest to równoznaczne z określeniem adresu głowy listy w pamięci. Organizację listy stosowej pokazano schematycznie na rys. 3.



Rys. 3. Organizacja listy stosowej

Ze względu na to, że na wartość W elementu listy przeznaczona jest jedna komórka pamięci maszyny cyfrowej, w każdym elemencie listy przechowywać można liczbę całkowitą (typu INTEGER), wartość logiczną (typu BOOLEAN) albo część liczby rzeczywistej (typu REAL) zajmującej dwie komórki pamięci. Każdy element listy zawierać może również dane upakowane (zostanie to wyjaśnione w drugiej części opracowania).

System procedur umożliwiający tworzenie list stosowych oraz wykonywanie operacji na elementach tych list, w programach napisanych w języku ALGOL 1900, składa się z dwu części:

- pakietu procedur podstawowych,
- pakietu procedur pomocniczych.

Pakiet procedur podstawowych zawiera 9 procedur o następujących nazwach:

GENERATE,	LOSE,
NIL,	GIVE,
PRESERVE,	STORE,
RENOVATE,	NULL.
SAVE,	

Oprócz nich, w skład pakietu procedur podstawowych wchodzi pewna liczba procedur systemowych, niedostępnych dla użytkownika poprzez program napisany w ALGOL-u.

W skład pakietu procedur pomocniczych wchodzą następujące procedury użytkowe:

BOININT	INT
REININT	BOOL
INTINBO	REAL 1
INTINRE	REAL 2
SETI	PACK (TPACK)
SETB	UNPACK (TUNPACK)
SETR1	FIXFLOAT
SETR2	FLOATFIX
	MASK (TMASK)

Aby utworzyć w programie w ALGOL-u dowolne listy stosowe, trzeba na początku odpowiedniego bloku programu opisać jednowymiarowe, dwuelementowe tablice typu INTEGER, których przyjęte nazwy będą stanowiły oznaczenia list. Wyrażenia graniczne opisu tych tablic przyjmuje się równe 0:1. Zanim rozpocznie się korzystanie z listy stosowej trzeba do drugiego miejsca tablicy odpowiadającej danej liście wpisać symbol NIL. Do tego celu służy bezparametrowa funkcja NIL.

Oprócz tego należy jeszcze ustalić pewne wewnętrzne warunki początkowe programu, który będzie korzystał z list stosowych. Generowanie warunków początkowych odbywa się globalnie dla całego programu, tylko jeden raz przed użyciem list niezależnie od ich liczby i sposobu ich rozlokowania w blo-

kach. Musi to nastąpić koniecznie przed wykonywaniem jakiejkolwiek operacji na elementach list (poza początkowym zapisem symbolu NIL). Służy do tego celu bezparametrowa procedura GENERATE. Jej zadaniem jest między innymi generowanie listy FREE LIST niedostępnej dla użytkownika, a zezwalającej na ekonomiczną gospodarkę pamięcią przydzieloną do programu.

Przykładowo utworzenie, w programie dwóch list LX i LY wymaga umieszczenia w tym programie następujących zapisów:

```
:
BEGIN
  INTEGER ARRAY LX[Ø : 1];
  :
  LX [1] : = NIL;
  GENERATE;
BEGIN
  INTEGER ARRAY LY [Ø : 1];
  :
  LY [1] : = NIL;
```

Opisane w programie dwuelementowe tablice stanowią głowy poszczególnych list. Są one bezpośrednio dostępne w programie. W miejscu o wskaźniku Ø przechowuje się wartość głowy, natomiast w miejscu o wskaźniku 1 - adres następnego elementu listy lub symbol NIL, gdy lista jest tylko jednoelementowa, a więc jej ogon jest pusty. Umożliwia to bezpośredni zapis wartości do głowy każdej listy za pomocą instrukcji podstawienia, np. LX[Ø]: = W;

Odczytu wartości głowy listy dokonuje się przez zapis nazwy tablicy reprezentującej głowę listy ze wskaźnikiem Ø w dowolnym wyrażeniu arytmetycznym, np. Z : = LX [Ø] + 2;

Należy zwrócić uwagę na to, że poza wstępnym zapisem symbolu NIL w miejsce o wskaźniku 1, nie należy więcej tego miejsca używać, a w żadnym wypadku nie wolno dokonywać do tego miejsca zapisu wartości. Grozi to bowiem dezorganizacją programu.

Podstawowe operacje na elementach list wykonuje się za pośrednictwem procedur PRESERVE oraz RENOVATE.

Procedura PRESERVE służy do przesunięcia wartości przechowywanej w głowie listy do pierwszego elementu ogona. Wartość głowy nie ulega przy tym zmianie. Nie ulegają również zmianie poprzednie wartości elementów ogona, lecz ogon wydłuża się o jeden element. Oznaczając symbolami I_i wartości poszczególnych elementów listy stosowej, operację tę można przedstawić następująco:

$I_0 \mid \underbrace{I_1, I_2, \ldots\ldots I_N}_{\text{ogon}}$ - postać listy przed użyciem procedury PRESERVE

głowa

$I_0 \mid I_0, I_1, I_2, \ldots I_N$ - postać listy po użyciu procedury PRESERVE

Parametrem procedury PRESERVE jest zmienna ze wskaźnikiem Ø o nazwie reprezentującej daną listę. Chcąc przykładowo zapisać do pustej listy LX dwie wartości 5 oraz -3, można tego dokonać za pomocą następującej sekwencji instrukcji:

```
LX [Ø] : = 5 ;
PRESERVE (LX [Ø]);
LX [Ø] : = -3 ;
```

Po wykonaniu tych instrukcji zawartość listy LX będzie następująca:

$$-3 \mid 5$$

Operacji odwrotnej do wyżej opisanej dokonuje się za pomocą procedury RENOVATE. Użycie tej procedury powoduje przepisanie wartości przechowywanej w pierwszym elemencie ogona listy do jej głowy i następnie usunięcie z listy pierwszego elementu ogona.
Lista zostaje więc skrócona o jeden element. Poprzednia zawartość głowy listy zostaje zniszczona.
Schematycznie operację tę można przedstawić następująco:

$I_0 \mid I_1, I_2, I_3 \ldots\ldots I_N$ - postać listy przed użyciem procedury RENOVATE

$I_1 \mid I_2, I_3, \ldots\ldots I_N$ - postać listy po użyciu procedury RENOVATE

Procedura RENOVATE wymaga dwóch parametrów. Pierwszy z nich to zmienna ze wskaźnikiem Ø o nazwie reprezentującej listę, drugi natomiast parametr jest etykietą. Jeżeli użyta zostanie procedura RENOVATE, a ogon listy był pusty, wówczas postać ani zawartość listy nie ulegną zmianie, a wyjście z procedury nastąpi skokiem do podanej etykiety.

Przykładowo, jeśli zostanie użyta procedura RENOVATE w odniesieniu do listy utworzonej jak w poprzednim przykładzie, to otrzymana w wyniku działania instrukcji procedury: RENOVATE (LX[Ø],ETYK); postać listy LX będzie następująca: 5|
Powtórne wywołanie procedury w sposób podany wyżej nie wywoła żadnych skutków poza przejściem w programie do wykonywania instrukcji opatrzonej etykietą ETYK.

Wykorzystując procedury PRESERVE i RENOVATE można wykonywać dowolne operacje na elementach list stosowych. W niektórych jednak przypadkach używanie tych procedur jest mało wygodne. Z tego względu wprowadzono dodatkowo procedury: LOSE, GIVE, STORE.

Procedura LOSE służy do usuwania z listy pierwszego elementu jej ogona. Głowa listy pozostaje niezmieniona, natomiast ogon listy ulega skróceniu. Schematycznie, można to przedstawić następująco:

$I_0 \mid I_1, I_2, I_3, \ldots\ldots I_N$ - lista przed użyciem procedury LOSE

$I_0 \mid I_2, I_3, \ldots\ldots I_N$ - lista po użyciu procedury LOSE

Procedura LOSE zawiera dwa parametry o takim samym znaczeniu, co w przypadku procedury RENOVATE.

Procedura GIVE służy do odczytu wartości dowolnego elementu ogona listy. Użycie tej procedury nie zmienia ani postaci, ani zawartości listy. Procedura ta wykonana jest jako funkcja, co oznacza, że odczytana wartość jest podstawiona pod nazwę procedury. Posiada ona trzy parametry: zmienną ze wskaźnikiem Ø reprezentującą nazwę listy, etykietę, do której wykonywany jest skok w przypadku gdy zażąda się wartości elementu nie należącego do listy oraz wartość określającą numer elementu ogona listy, z którego ma być odczytana zawartość. Jeżeli w wywołaniu procedury podany zostanie numer nieistniejącego elementu listy, to poza wyjściem skokiem do podanej etykiety, wartością funkcji GIVE będzie długość ogona listy.

Jeżeli przykładowo lista LX ma postać: -3|5, 1,7, -2, wówczas wynikiem działania instrukcji:

I : = GIVE (LX [Ø], ETYK, 3);

będzie nadanie zmiennej I wartości 7. Natomiast wykonanie instrukcji:

I : = GIVE (LX [Ø], ETYK, 5),

spowoduje przejście do instrukcji opatrzonej etykietą ETYK,a wartość zmiennej I będzie równa 4 (długość ogona listy).

Procedura STORE spełnia odwrotne zadanie co procedura GIVE. Za jej pośrednictwem można dokonać zapisu wartości dla dowolnie wybranego elementu listy. Procedura ta jest również wykonana jako funkcja, a jej wartością będzie poprzednia wartość wybranego elementu ogona listy. Procedura STORE posiada 4 parametry, z których trzy pierwsze mają to samo znaczenie co w procedurze GIVE, natomiast czwarty jest wartością, która ma być zapisana w wybranym elemencie listy.

Rozpatrując listę z poprzedniego przykładu, wykonanie instrukcji: I := = STORE (LX [Ø], ETYK, 2 , 33), spowoduje zapisanie w drugim elemencie ogona listy wartości 33 i nadanie zmiennej I wartości 1, czyli poprzedniej wartości drugiego elementu ogona listy. Po wykonaniu instrukcji postać listy będzie następująca:

-3 | 5, 33, 7, -2.

Procedury służące do odczytu zawartości elementów list sprawdzają każdorazowo czy ogon listy nie jest pusty. Sprawdzenia tego można również dokonać, stosując w programie funkcyjną procedurę NULL, typu BOOLEAN. Parametrem typu INTEGER tej procedury jest zmienna ze wskaźnikiem Ø o nazwie reprezentującej listę stosową. Wartością procedury NULL jest TRUE, gdy ogon badanej listy jest pusty oraz FALSE w przeciwnym razie.

Niezmiernie ważną sprawą, szczególnie w systemie wieloprogramowym, jest racjonalna gospodarka pamięcią. Aby zadośćuczynić jej wymogom, do systemu wprowadzono dodatkową listę tzw. nieużytków, FREE LIST. Jest ona niedostępna dla użytkowników, natomiast korzystają z niej procedury systemu. Listę tę tworzy procedura GENERATE i w chwili jej tworzenia jest ona listą pustą.

Do tworzenia i rozszerzania list użytkowych służy omówiona wcześniej procedura PRESERVE. Każda próba dołączenia do dowolnej listy, nowego elementu za pomocą procedury PRESERVE, wywołuje w pierwszej kolejności sprawdzenie czy element ten nie może pochodzić z listy FREE LIST. Jeżeli tak, to pierwszy element listy nieużytków zostaje dołączony do rozszerzanej listy, natomiast lista FREE LIST zostaje skrócona o ten element. Gdy natomiast lista nieużytków jest pusta, procedura PRESERVE tworzy nowy element rozpatrywanej listy z komórek znajdujących się bezpośrednio poza aktualnym obszarem roboczym programu.

Skrócenie dowolnej listy za pośrednictwem procedur LOSE i RENOVATE powoduje dołączenie usuniętego elementu tej listy do listy nieużytków, która w ten sposób zostaje wydłużona.

Pewne komplikacje w takim sposobie korzystania z pamięci wywołuje blokowa struktura programów napisanych w języku ALGOL. Każde bowiem otwarcie bloku powoduje nowy przydział pamięci programowi, natomiast zamknięcie bloku wywołuje cofnięcie tego przedziału. Ponieważ zasada kompilacji programów nie została naruszona, w związku z tym cofanie przydziału pamięci następuje także wtedy, gdy w obszarze pamięci, który w dalszym ciągu staje się niedostępny dla programu, znajdują się elementy aktywnych list (w bloku została użyta procedura PRESERVE). Wynika stąd potrzeba "ratowania" obszarów pamięci, którym w momencie zamykania bloków został cofnięty przydział, a w których znajdują się elementy list. Do tego celu służy bezparametrowa procedura SAVE. Instrukcję tej procedury umieszcza się bezpośrednio po symbolu END kończącym dany blok.

Jeżeli w obszarze pamięci związanym z tym blokiem nie znajdują się elementy żadnych list, wówczas instrukcja procedury SAVE nie wywołuje żadnych zmian. W przeciwnym razie następuje ponowny przydział pamięci w taki sposób, że obejmuje on wszystkie elementy list, a miejsca pamięci w tym obszarze nie zajmowane przez elementy list zostają przeniesione do listy nieużytków.

Pewien problem powstaje w wypadku, gdy w programie znajduje się instrukcja skoku prowadząca na zewnątrz bloku. Należy wtedy tak zmienić treść programu, by przejście na zewnątrz bloku następowało zawsze przez symbol END kończący blok.

Przykładowo, jeśli wymagany jest następujący układ programu:

```
        :
        E : ......
        :
        BEGIN                                  ┐
         :                                     │
         PRESERVE (X [Ø]);                     │   blok programu
         :                                     │
         IF  B  THEN  GO  TO  E ;              │
         :                                     │
         END ;                                 ┘
         :
```

należy postąpić następująco:

```
        :
        E  : .....
        :
        BEGIN
        :
        PRESERVE (X [Ø]);
        :
        IF  B  THEN  GO  TO  E1 ;
        :
        E1 : END ;
             SAVE ;
             IF  B  THEN  GO  TO  E;
```

Opisana wyżej organizacja gospodarki pamięcią pozwala w maksymalny sposób wykorzystać obszary pamięci, nie pozostawiając w pamięci żadnych "dziur" i równocześnie zapewnia traktowanie elementów list na równi ze zmiennymi programu.

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.

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

Часть 1. Описание системы и базовые процедуры

Р е з ю м е

В статье представлено описание системы процедур служащих обработке столбовых списков на языке Алгол 1900. Эта система разработанная в Силезском Политехническом Институте.

Рассмотрено общую концепцию системы, структуру обрабатываемых объектов и представлено описание базовых процедур, при помощи которых есть возможное образование и использование столбовых списков.

THE SYSTEM OF PROCEDURES FOR PUSHDOWN LISTS
PROCESSING IN ALGOL 1900

Part. 1. Description of the system and basic procedures

S u m m a r y

The paper includes a description of the system of procedures for pushdown lists processing. The system has been elaborated in Silesian Institute of Technology. The general idea of the system the structure of the processed objects, as well as the description of the basic procedures which enable composing of pushdown lists and using them are represented.